基于 LSM TREE 的键值存储项目实验报告

朱子墨 523031910852

2025 年 3 月 29 日

1 背景介绍

LSM-Tree（Log-Structured Merge-Tree）是一种适用于高吞吐量写入的存储结构，由 Patrick O’Neil 等人于 1996 年在论文中首次提出。 主要用于数据库（如 LevelDB、RocksDB）和键值存储系统（如 Cassandra）。它通过批量写入和分层合并（Compaction）来优化磁盘 I/O，从而提高写性能。

LSM Tree 尤其适合写密集型场景（如日志存储、物联网设备数据、时序数据库），其设计哲学体现了” 写优化” 与” 读优化” 的权衡，成为大数据时代高性能存储系统的基石之一。

2 性能测试

2.1 实验基本配置

为了评估 LSM-Tree 键值存储系统的性能，我先完善了基本的代码的设计，在通过了 correctness 和 persistence 两个测试后，仿照 correctness 测试的代码设计了性能测试文件，分别测试 PUT、GET 和 DEL 三种基本操作的吞吐量和平均时延。测试运行环境如下：

实验环境要素	配置
操作系统	Windows 11(26100.3194)
CPU	Intel Core i7-1360P(2.20 GHz)
RAM	16GB

表 1: 实验基本环境配置

测试一共进行 3 组，每一组测试先使用 PUT 操作插入一定数量的键值对，再使用 GET 操作进行查询，最后使用 DEL 操作进行删除。每组用于测试的字符串长度范围为 [1,100]，平均字符串长度为 50，3 组测试分别进行了 20000，100000，200000 次操作。

在某个操作开始前，记录系统时间；在操作结束后，记录系统时间；利用系统时间差除以操作数量，即可得到该操作的吞吐率；取倒数则得到操作的平均时延。

2.2 实验初步分析与预期结果

基于代码完善过程以及 LSM-Tree 的设计特性，我对测试结果作出如下分析和预期：

- 三种操作均可能能受到测试集大小影响，当测试集过小时，可能无法发挥 sstable 的作用，导致测试结果不准确。测试集很大时可能会造成磁盘存储空间不足，影响系统性能。因此，我选择了相对适中的字符串长度和字符串数量进行测试。
- 由于 LSM TREE 是面向密集型写入的存储结构，因此 PUT 操作可能会性能优于 GET 和 DEL；GET 和 DEL 均需要进行搜索，而且会访问并查找磁盘文件，因此 GET 和 DEL 操作的效率会比较接近，且低于 PUT 操作。

2.3 实验结果与分析

通过运行性能测试，我得到了三种操作的性能数据，如下所示。其中，吞吐率保留两位小数，数据单位为 (操作数/秒)；平均时延保留一位小数，数据单位为毫秒。

操作类型	size=20000	size=100000	size=200000
PUT	54129.69	112413.10	58645.87
GET	8760.75	1790.68	782.09
DEL	9251.72	771.96	460.17

表 2: LSM-Tree 键值存储系统吞吐率性能测试结果 (单位：(ops/sec))

操作类型	size=20000	size=100000	size=200000
PUT	0.018	0.009	0.017
GET	0.114	0.558	1.279
DEL	0.108	1.295	2.173

表 3: LSM-Tree 键值存储系统平均时延性能测试结果 (单位：(ms))

通过表格有以下发现：

- PUT 操作的吞吐率远高于 GET 和 DEL 操作。这可能是由于 PUT 操作对于内存的访问较少导致的。测试数据的写入一般在跳表中完成，仅在超出文件大小限制时才进行耗时较多的 compaction 合并操作。相比较访问 Cache 而言，访问磁盘的 I/O 会带来更多的时间损耗。

- GET 操作整体上吞吐率远低于 PUT 操作，但高于 DEL 操作；GET 在查找时查询时需访问磁盘的 SSTable 信息，导致随机 I/O 开销大。而 DEL 操作实际上是一次 GET 操作（查询存在性）加上一次 PUT 操作（设置 DELETE 标签），因此时间性能比 GET 更差。
- GET 和 DEL 操作在测试集数据量相对较大的情况下性能会比较差。这是由于 GET 操作的查找时间随着数据数量的增加而显著增加；DEL 操作包含 GET 操作，因此吞吐率也会随着数据数量的增大而降低。

3 结论

通过实现和测试基于 LSM Tree 的键值存储系统，验证了其核心特性：

- 读写性能较好：PUT 和 GET 的吞吐率较高，特别是 PUT 写入操作性能优异，这符合 LSM TREE 的批量化、大规模写入优势。
- 删除成本高：DEL 性能最低，删除记录的时间开销很大。
- 在较大数据规模情况下 PUT 操作仍具有优异性能，这符合 LSM TREE 大规模批量化写入的优势。

4 致谢

在完成此实验过程中，GitHub 上的相关开源实现也给了我很多启发，如 ChatGPT 和 deepseek 等 AI 工具也给予了很多帮助。此外，杨宇峰同学在关于实验配置和测试方面也给予了我建议和帮助。

5 其他和建议

在实验过程中，我遇到了一些挑战和困难。首先是合并操作的 compaction 函数，在完成后出现了一些 bug，经过检查发现一些边界情况没有充分全面考虑，造成了错误。为此，我增加了判断函数和检查逻辑以提高程序的健壮性。同时，测试函数的设计过程中最开始出现了错误，测试集和测试函数的设计也需要仔细考虑，确保程序测试结果的合理性。

建议：- 补充更多的单元测试和集成测试，增加分项测试 - lab 文件夹中的 README 文件说明较为简略，建议补充一些细节，能够更方便地配置环境和测试