LSM-KV 项目报告

闫景升 521030910309

2023 年 3 月 29 日

1 背景介绍

LSM-KV 是一种具有性能特征的数据结构，可以为具有高插入量的文件（例如事务日志）提供索引访问数据。LSM 树和其他搜索树一样，维护键值对。LSM 树将数据保存在两个或多个独立的结构中，每个结构都针对其各自的底层存储介质进行了优化；数据在两个结构之间有效地、批量地同步 [1]。

2 测试

2.1 性能测试

2.1.1 预期结果

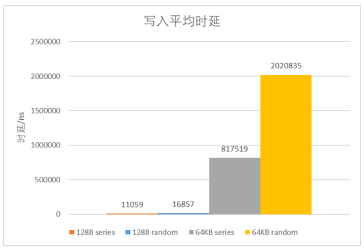
1. 缓存预测：缓存 Bloom Filter 和索引的结构时延最低，只缓存索引的结构其次，不缓存任何内容的结构时延最长，且远大于前两者

2. Compaction 影响预测：不断插入的情况下，每插入一定量的数据，便会进行合并，表现出 Put 吞吐量的骤降

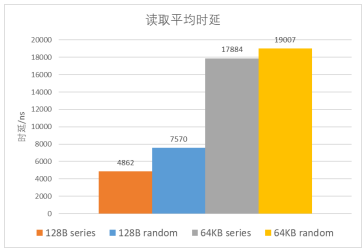
3. Leveling 配置影响预测：Leveling 偏多的情况下，读取吞吐量较大，写入吞吐量较小。Tiering 偏多的情况下，写入的吞吐量较大，读取吞吐量较小。采用混合策略，较低的层采用 Tiering，较高的层采用 Leveling，可以达到读写吞吐量均为较优水平的情况。

2.1.2 常规分析

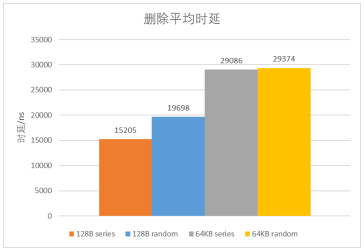
1. 在 key 的大小均为（8B），分别对 value 为小型数据（128B）和大型数据（64KB），在顺序操作和乱序操作下，分别在总数据量 32MB、64MB、128MB 的情况下测试 Get、Put、Delete 操作的延迟，并计算出各组的平均延迟，实验结果如图 1。



(a) Put Latency



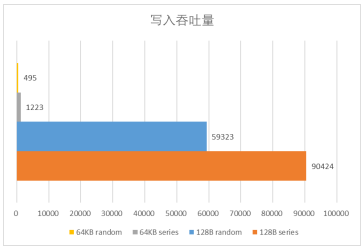
(b) Get Latency



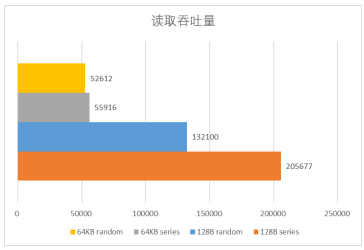
(c) Delete Latency

图 1: 操作平均时延

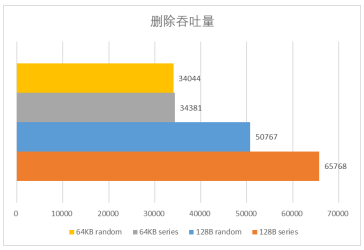
2. 在 key 的大小均为 8B，分别对小型数据（128B）和大型数据（64KB），在顺序操作和乱序操作下，分别在总数据量 32MB、64MB、128MB 的情况下测试 Get、Put、Delete 操作的吞吐量，并计算出各组的平均吞吐量，实验结果如图 2。



(a) Put Capacity



(b) Get Capacity



(c) Delete Capacity

图 2: 操作吞吐量

2.1.3 索引缓存与 Bloom Filter 的效果测试

需要对比下面三种情况 GET 操作的平均时延在 key 为 8B，value 为 128B，总数据量分别为 16MB，32MB，64MB 的情况下，分别采用以下三种缓存策略，在顺序读取和乱序读取的情况下对 Get 操作的平均时延进行测试，实验结果如图 3

1. 内存中没有缓存 SSTable 的任何信息，从磁盘中访问 SSTable 的索引，在找到 offset 之后读取数据

2. 内存中只缓存了 SSTable 的索引信息，通过二分查找从 SSTable 的索引中找到 offset，并在磁盘中读取对应的值

3. 内存中缓存 SSTable 的 Bloom Filter 和索引，先通过 Bloom Filter 判断一个键值是否可能在一个 SSTable 中，如果存在再利用二分查找，否则直接查看下一个 SSTable 的索引

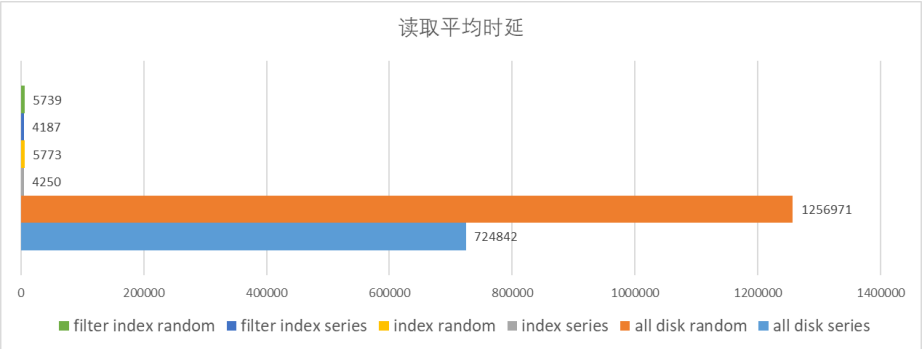


图 3: 不同缓存对 Get 时延的影响

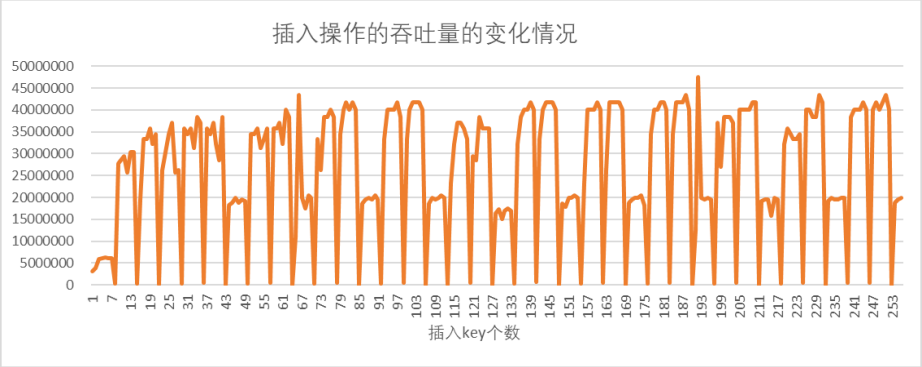


图 4: 不断插入数据的情况下,Put 吞吐量变化

2.1.4 Compaction 的影响

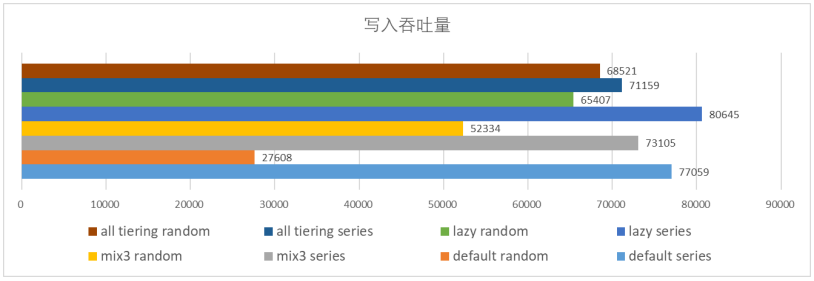
不断插入数据的情况下，统计每秒钟处理的 PUT 请求个数（即吞吐量），并绘制其随时间变化的折线图，如图 4

2.1.5 Level 配置的影响

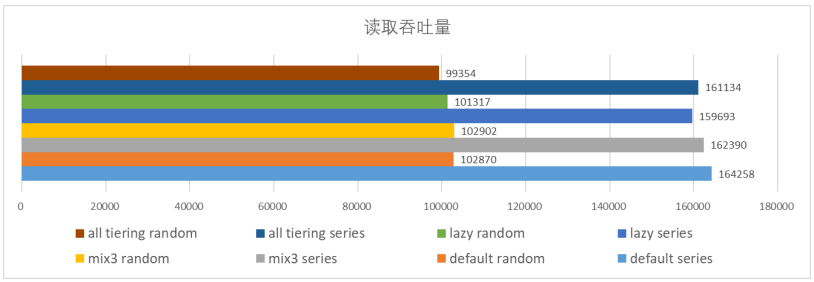
对每层是 Leveling 还是 Tiering 进行改变，分别按以下配置规则，测试其对存储系统 Put、Get、Delete，Range（查找指定范围内的所有值）分别在顺序和乱序情况下吞吐量的影响，实验结果如图 5。

1. All Tiering：所有层均为 Tiering
2. Mix3：前三层为 Tiering，后面均为 Leveling
3. Lazy：当前最后一层为 Leveling，其余层为 Tiering

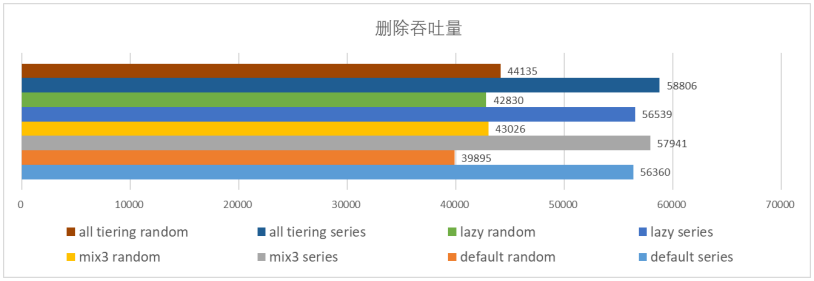
4. Default：只有第一层为 Tiering，其余层为 Leveling



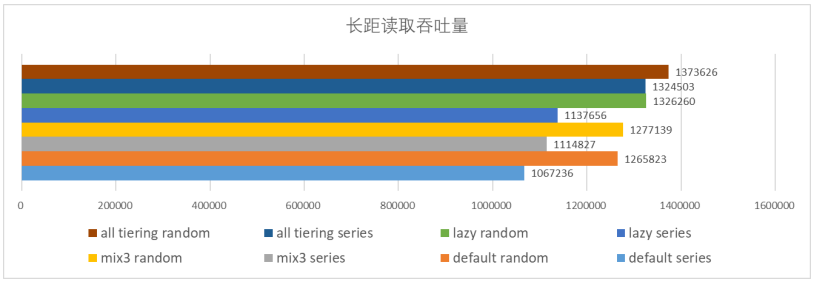
(a) Put Capacity



(b) Get Capacity



(c) Delete Capacity



(d) Range Capacity

图 5: 不同设置对吞吐量的影响

3 结论

1. 基本完成 Project 预定目标，完成搭建 LSM-KV 结构。
2. 实验结果基本符合预期。
3. 在内存中应缓存所有的索引和 Bloom Filter 以尽量提高效率。
4. 配置层次类型时，应按照实际情况进行选择。写入更多时，采取更多的 Tiering 策略，读取更多时，采取更多的 Leveling 策略。而要达到读写均衡效果，应采取混合策略，较低的层采用 Tiering，较高的层采用 Leveling。

4 致谢

1. 感谢知乎文章 LSM Tree 的 Leveling 和 Tiering Compaction[2]
2. 测试方面参考了 Google 的 levelDB [3]
3. 测试分析方面参考了 Dostoevsky [4]
4. 感谢刘洋同学的麦当劳支持

5 其他和建议

必须要有一个服务器，自己电脑跑这玩意太心疼了。。。。而且还慢，等数据等好久好久好久好久。

最后被文件名重复后覆盖的问题折磨了很久，拖了好几天，然后改了一行代码就解决了。。。

总体还是非常好的一个项目，锻炼了自己的 C++ 能力，对数据库项目也有了更多了解。之前都是在运行速度在秒级别的程序，终于接触到了大数据和长运行时间。

参考文献

[1] `https://en.wikipedia.org/wiki/Log-structured_merge-tree`.

[2] `https://zhuanlan.zhihu.com/p/112574579`.

[3] `https://github.com/google/leveldb`.

[4] `https://nivdayan.github.io/dostoevsky.pdf`.