基于交错磁记录的 KV 存储综述

翟明晗

摘要： 作为冯·诺依曼体系五大组成部分之一的存储器，广泛应用于计算机、物联网、消费电子、云存储、网络存储等重要领域，是一种极为基础和重要的产品。随着 5G 时代的到来，数据爆炸式增长，移动通信、物联网等领域对存储器的需求量迅速增加，且对存储容量、功耗、读写速度、使用寿命等性能指标要求愈来愈高。近年来已出现多种新型存储器，极大的提升了存储器的各项性能，如 SMR 叠瓦式磁记录、交错磁记录等新型硬盘驱动器，突破了存储介质的容量极限，具有大容量低功耗等特点。这些存储器可针对基于 LSM 合并树的 KV 键值存储设计专门的固件来提升 KV 存储读写效率，极大的减少读写放大问题，提供了更好的数据读写性能。本文针对新型存储器和在其之上对基于 LSM 合并树的 KV 存储提出的优化算法进行介绍。

关键词：键值存储；新型存储器；交错磁记录

A Survey of Key-Value Store Based On Interlaced Magnetic Recording

Minghan Zhai

Abstract: As one of the five components of the von Neumann system, memory is widely used in computer, Internet of Things, consumer electronics, cloud storage, network storage and other important fields. It is a very basic and important product. With the advent of the 5G era, data explosion growth, mobile communication, Internet of Things and other areas of memory demand is rapidly increasing, and storage capacity, power consumption, read and write speed, service life and other performance indicators are increasingly demanding. In recent years, there have been a variety of new memory, greatly improve the performance of memory, such as SMR imbricate magnetic recording, interlaced magnetic recording and other new hard disk drives, breaking through the capacity limit of storage media, with large capacity and low power consumption characteristics. These memories can be designed for KV storage based the LSM merge tree with special firmware to improve the read and write efficiency of KV storage, greatly reduce the read and write amplification problem, and provide better data read and write performance. This paper introduces the new memory and its optimization algorithm based on KV storage based the LSM merge tree.

Keywords: Key Value storage; New memory; Interlaced magnetic recording

1 引言

传统的存储器多为CMR硬盘驱动器,CMR称为传统磁记录方式，这种方式保留了最早PMR替代LMR时的传统技术，即：磁道间留有保护间距，数据不会被重复叠写。传统的磁盘读写磁头宽度不一致，写磁头比读磁头更宽，为了防止数据被破坏，每条磁道间需要保持一定的安全距离，因此虽没有写放大问题，但由于其超顺磁效应，在提供更高的面密度以降低每 GB 硬盘成本方面已经达到瓶颈，整个磁面密度相对也较低，当数据需求量爆炸式增长时，该类磁盘的性能早已不满足需求。在各种可以打破这种面密度限制的新技术中，SMR技术是CMR的合格代替方案，可以在有限的变化下提供更高的面密度增益。SMR被称为叠瓦式磁记录，是有重叠的磁记录方式，磁面上的每条磁道都会和邻接的上一条磁道重叠一部分，而数据统一写在磁道的一侧，SMR的磁瓦结构充分利用了磁道间的安全间隙，使得磁盘的存储密度增加了，存储相比CMR提高25%，但也带来了一些问题，由于SMR写磁头操作限制，使得SMR更加倾向于顺序写，随机写的性能很糟糕，随机写需要将下一条磁道相应位置的数据复制到缓冲区，以便之后进行写回操作，因此具有较长的写尾延迟，在长时间写入数据的时候，可能会出现速度急剧下降的情况。SMR需要较大的缓存来进行写回操作，并且降低了转速，且有掉速、发热、噪音和寿命低等问题，因此SMR适合应用于数据的归档和备份，作为仓库盘使用，而不适合 IO 较多的用户盘。近年来出现了名为IMR的新型存储模式，已经被验证为比SMR有着更大的面密度和更少的磁道重写问题，使IMR成为CMR更理性的继承者，IMR被称为交错磁记录，比SMR有着更高的存储密度，存储相比CMR提高了40%，它缓解了SMR中随机写带来的写放大问题,IMR的磁道分为上下两层，两层相互交错，下两层的磁道被两个相邻的上层磁道部分覆盖，下磁道较宽，上磁道较窄，下层磁道未被覆盖的部分就是读磁头的宽度。传统的PMR技术无法控制磁道宽度，只能在HAMR或MAMR上实现IMR（使用激光或微波的方式），控制不同的宽度，IMR上下层磁道的存储密度和数据传输速率因此也是不一样的。对IMR的上层磁道的写操作不会引起写放大问题，对下层磁道的写操作会引起相邻的两个上层磁道的写放大问题，导致两次多余的读和写，总的来说，IMR的随机写放大问题比SMR小的多。

基于 LSM-tree （ Log-Structured-Merge-Tree）的KV（Key-Value）存储可以在HDD（Hard Disk Drive）上提供高吞吐量的写密集型应用程序。近年来，新出现的交错磁记录（IMR）技术使得基于交错磁记录（IMR）的硬盘驱动器由于其高面密度而成为一种具有高成本效益的 KV 存储器的理想选择。然而，在基于IMR的HDD上部署基于LSM-tree的KV存储可能会在随机读写的吞吐量上有着明显的下降，这是因为基于IMR的HDD的RMW（Read-Modify-Write）过程可能会放大基于LSM-tree的KV存储的压缩过程引入的后台I/O，这种放大的后台I/O可能会降低压缩过程的效率，进一步降低随机读写的吞吐量。此外，以往的IMR磁道分配算法并不能很好的弥补这种吞吐量下降缺陷，因为基于LSM-tree的KV存储的压缩过程背后隐藏着特殊的软件行为因素，因此针对该问题，近年来也出现了一些新的针对基

于 LSM-tree 和 IMR 的 KV 存储优化算法以及中间件，如 KVIMR 中间件和其内置的磁道分配算法。KVIMR 极大的提高了 KV 存储随机读写的吞吐量，并且保持了数据崩溃一致性，相比于当前基于 IMR 硬盘最先进的磁道分配算法，KVIMR 在写密集型任务下，将整体吞吐量提高了 1.55 倍，在 HDD 空间充足情况下，更是将整体吞吐量提高了 2.17 倍。

本文第二节对 IMR 交错磁记录和基于 LSM-tree 树进行了简要介绍，并在第三节对近年来最新且最先进的技术方案进行详细的讨论。最后第四节对本文工作进行总结。

2 原理和优势

2.1 IMR 交错磁记录

IMR（Interlaced magnetic recording）交错磁记录是一种新的磁盘存储模式，它采用“交错”的磁道布局，如图 1c 所示，通过缩短相邻磁道之间的距离来增加磁道的面密度。基于 IMR 的 HDD 磁道分为上层磁道和下层磁道，每条底部磁道被两个相邻的上层磁道部分覆盖，如同 CMR 和 SMR 的写放大问题，对基于 IMR 的 HDD 任何下层磁道的写操作都会破坏相邻的两条上层磁道相应位置的有效数据。因此，为了保护上层磁道的有效数据在写入下层磁道时不会被破坏丢失，基于 IMR 的 HDD 需要专门分配一块缓冲区，用于暂存上层磁道的有效数据，并在之后写回到上层磁道（基于 IMR 的 HDD 任何上层磁道写操作都不会引发写放大问题）。SMR 叠瓦式磁记录的磁道分布如图 1b 所示，它的主要目的是提高存储密度，且相对 CMR 提高了 25%，但其随机读写性能非常低，SMR 牺牲了随机写能力，修改磁道上的数据时需要对所

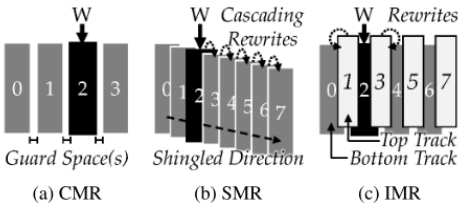

图 1 CMR、SMR 和 IMR 磁道分布图

有下游磁道进行重写，因此存在十分严重的写放大问题，且需要较大的缓冲区去缓存重写时的下游磁道的有效数据，除此之外 SMR 还不得不降低转速，更是降低了 I/O 性能。不过近年来 SMR 也出现了一些优化算法和相应的产品，如将 SMR 磁盘的磁道划分为多个磁道带，磁道带之间保持安全间隙（但磁道带不宜过大，太小又不能有效提高存储密度），目前市面上存在一些 SMR 的衍生产品，如 DM-SMR（磁盘管理式 SMR）、HM-SMR（主机管理式 SMR）、HA-SMR（主机感知式 SMR）等，SMR 的主要研究方向有写放大问题、数据清除策略和 SMR 磁盘的上层应用（如文件系统：SMRfs、HiSMRfs）等。相比于 SMR，IMR 有着更高的存储密度，相比 CMR 提高了 40%，它能够大大缓解 SMR 中随机写来的写放大问题，由于该技术较新，目前市面上并没有实际的衍生产品。IMR 磁道分为顶层磁道和底层磁道，两层相互交错，底层磁道被两个顶层磁道部分覆盖，中间没有覆盖的为读磁头宽度。其中底层磁道需要较高强度的激光或者微波才能写入数据：而顶层磁道写入数据所需要的激光或微波强度较低，不会破坏底层磁道的有效数据。IMR 的分配方式主要为两阶段和三阶段分配法，两阶段分配法为先顺序分配底层磁道，底层磁道分配完后顺序分配顶层磁道，三阶段分配法的第一阶段和两阶段法一致，在第二阶段每两个顶层磁道分配一条磁道，第三阶段则分配剩余的顶层磁道，这种分配方式在前两阶段大大

减少了写放大问题，但在第三阶段写放大问题依旧严重。IMR 的主要研究方向为数据更新方式、数据写缓冲区、冷热数据交换和 IMR 磁盘的上层应用如基于 IMR 的文件系统、数据库系统和 RAID 系统等。

2.2 基于 LSM-tree 的 KV 存储

LSM-tree（日志结构合并树）是一个横跨内存和磁盘的，包含多颗子树的森林。LSM-tree 分为 level 0，level 1，...，level n 多颗子树，其中只有 level 0 在内存中，其余 level 1-n 在磁盘中。内存中的 level 0 子树一般采用排序树（红黑树/AVL 树）、跳表或者 TreeMap 等这类有序的数据结构，方便后续顺序写磁盘。磁盘中的 level 1-n 子树，本质是数据排好序后顺序写到磁盘上的文件，只是叫做树而已。每一层的子树都有一个阈值大小，达到阈值后会进行压缩合并，合并结果写入下一层。只有内存中数据允许原地更新，磁盘上数据的变更只允许追加写，不做原地更新操作。由于 HDD 提供了高写吞吐量，LSM-tree 激发了许多著名 KV 存储的诞生，如 RocksDB、LevelDB 和

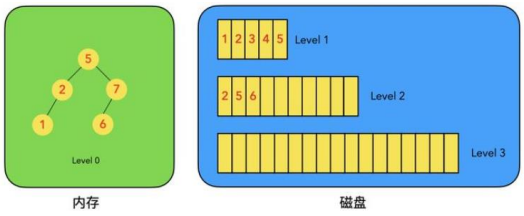

图 2 LSM-tree 结构图

HyperLevelDB。一般来说，这些基于 LSM-tree 的 KV 存储采用了类似的设计理念来管理 KV 键值对，并支持类似的一组 put、get、delete 操作。为了在 HDD 上提供高吞吐量的操作，基于 LSM-tree 的 KV 存储首先将插入的 KV 键值对在内存中一个名为 Memtable 的跳表中排序。当 Memtable 超过了它的内存大小限制（例如 RocksDB 中为 64MB），基于 LSM-tree 的 KV 存储会创建一个新的 Memtable 来容纳新插入的 KV 键值对，同时旧的 Memtable 会被转换成一个不可变的基于内存排序的跳表，即 Immutable Memtable，在后台，一个重要的线程接管了 Immutable Memtable 并将其持久化到 HDD 中作为一个名为 SSTable 的排序字符串表。此外，磁盘中的所有 SSTable 都被组织成多个级别（即 L0~Ln），每个级别的大小限制都以指数级别增加（例如 LevelDB 和 RocksDB 的底数都是 10）。此外，从 Immutable Memtable 转换成 SSTable 后通常放在 L0 级别所在的层，一旦任何 Level 层总大小超过了该层的大小限制，后台进程都将对该层以级联的方式进行压缩合并操作，并将其存储到 Level 更高的一层，并将旧的 SSTable 删除。除了 put 操作，基于 LSM-tree 的 KV 存储还支持 get 操作，读取指定的 KV 键值对。具体来说，基于 LSM-tree 的 KV 存储将从 Memtable、Immutable Memtable 和 SSTable 依次从低级别到高级别搜索请求的 KV 键值对，此外，基于 LSM-tree 的 KV 存储还支持删除操作，从 KV 存储中删除特定的 KV 键值对。

3 研究进展

对 KV 键值存储的研究方向有多种，现有的研究有关于 LSM-tree 优化方向的，有针对新型存储器进行相关设计与优化方向的。值得一提的是，这些方向并不是相互独立的，而是相辅相成的，任何单一的方向所能提升的效果都可能是有限的，而将多者结

合起来会带来 1+1>2 的提升，本文分别调研了针对新型存储器（IMR）方向的研究以及针对LSM-tree存储磁道分配策略方向的研究，并着重总结了二者结合后对性能提升的影响。

3.1 新型存储器

传统的CMR的特性上文已详细说明，本小节将着重分析不同的新型存储器的优缺点。SMR衍生产品主要分为三类：DM-SMR磁盘管理式SMR,HM-SMR主机管理式SMR，AM-SMR主机感知式SMR。其中DM-SMR主机感知式SMR主要是为了与现有系统兼容而设计。它引入了类似SSD中FTL的转换层——STL，并需要一个较大的持久缓存，用以缓存随机写所需要缓存的写回数据，并需要维护一个LBA-PBA的映射表。它的缺点是随着数据量的增加，映射方式和清除策略使得STL越来越复杂，严重影响性能。HM-SMR主机管理式SMR的设计目的是将SMR的管理全部放在主机端。它利用INCITS提供的ZBC和ZAC接口,向主机提供SMR磁盘内部数据分布信息，并由主机完成LBA到PBA的映射操作。它的缺点是没有持久缓存，因此不接受非顺序写操作,无法随机读写。HA-SMR主机感知式SMR的设计目的是结合DM-SMR和HM-SMR的优势。它有内置的STL和持久缓存，因此可以处理非顺序写操作，且可以向主机公开数据分布信息，以便主机更好的组织I/O请求，它内部存在两个写指针（顺序写和随机写）。

SMR目前主要的研究方向有三个：写放大问题、数据清除策略和SMR磁盘的上层应用。其中写放大问题产生的原因是随机写操作或者数据更新造成。它的处理方式大致分为三种。第一种是in-place update，它的原理是将修改数据所在的整个磁道带读入内存，内存中修改后，再将整个磁道带写回原来的位置，该方式的优点是不需要GC操作，也不需要维护映射表，缺点是此道的的大小不易控制，存储密度与写放大问题需要权衡。第二种是out-of-place update，它的原理是将更新的数据追加到新的位置，并使得原数据位置无效。一种可行的方案是将访问区域划分为一个E区域和多个I区域。E区域用来存储,I区域用来永久保存数据。所有写数据都先被缓存到E区域，然后在需要的时候再写回到对应的I区域。该方案的缺点是E，I区域必须都是顺序写，不能随机写，且E，I区域都需要垃圾回收，并需要维护一个LBA-PBA的映射表。第三种方式是混合方式（SMaRT），该方案的主要思路是将无效磁道暂时作为安全间隙，使上一个磁道可以就地更新。

SMR数据清除策略主要是持久缓存的清除策略，一般是FIFO，DM-SMR清除工作由STL利用磁盘的计算能力和资源完成，因此会严重影响性能。而HA-SMR借助主机的计算能力完成清除工作，主要有两种模式，一种是aggressive cleaning，系统空闲时按FIFO策略进行清除，另一种是lazy cleaning，持久缓存或映射表空闲空间较低时才进行清除工作，这种清除方式持续时间长，且清除过程中无法响应外界I/O请求，产生了一定时间的阻塞。SMR磁盘的上层应用如文件系统，目前比较先进的是SMRfs和HiSMRfs文件系统。

IMR目前的衍生产品如DM-IMR，它将磁道划分为多个磁道组，每个磁道组中采用三阶段分配方式，它有一个Top-Buffer，将每个磁道组中最后几个未分配的上层磁道作

为缓存区，并且可以缓存同一下层磁道数据块的多次修改，缓冲区大小为磁道组大小的2%，因为可以限制映射表的大小，所以可以保存到内存中，加速映射操作。DM-IMR 的缓存清除策略是顺序清除，该方式可以回收连续的存储空间，避免缓冲区碎片化。但 DM-IMR 中的 Top-Buffer 随着磁道组空间的使用率增长而动态减缩，越来越小，优势也会降低，频繁的数据清除将会导致性能下降。可以保持大小固定，作为永久缓冲区，但是就不能存储有效数据了。IMR 的研究方向主要有四个：数据更新方式，分为就地更新和异地更新两种，IMR 就地更新的代价比 SMR 小很多，可以按照二、三阶段分配方式进行。该方式可以改变阶段写入的方向（奇偶阶段方向不同），可以减少寻道时间，保持数据流的空间局部性，且可以像 SMR 一样，将 IMR 磁道划分为多个磁道带。异地更新可以将下层的更新操作，写入到上层磁道中的一个位置上，将下层原数据置为无效。该方式的缺点是引入了回收下层磁道无效数据块的开销，引入了频繁更新映射表的开销。IMR 的就地更新代价不高，所以需要权衡就地更新还是异地更新。数据写缓冲区，由于都是先分配下层磁道，所以当下层磁道已有数据更新时，可以选择上层空闲磁道作为临时缓冲区（如 DM-IMR）。冷热数据交换，由于 IMR 上层磁道的写操作不会引起写放大，可以将下层磁道中的热数据与上层磁道的冷数据进行置换。这样数据更新操作在上层磁道完成，不需要额外的读写操作，不过上下层磁道容量不一样，不能简单置换磁道，由此引发的数据交换单位和管理开销有待研究。IMR 磁盘的上层应用研究，如基于 IMR 的文件系统，数据库系统和 RAID 系统等。

3.2 磁道分配策略

基于 LSM-tree 的 KV 存储面临的写放大问题不容忽视，由于 LSM-tree 的 compaction 策略，导致基于 LSM-tree 的 KV 存储必然面临着写放大问题。除了提高基于 HDD 的新型存储模式的吞吐量，还可以从基于 LSM-tree 的 KV 存储底层磁道分配策略着手，优化磁道分配策略，来减少写放大次数，进而提升整体吞吐量。近年来除了二、三阶段磁道分配策略，出现了一些新的磁道分配策略，如果 KVIMR 磁道分配策略。KVIMR 是位于基于 LSM-tree 的 KV 存储和基于 IMR 的 HDD 之间的中间件，提供了一个 POSIX 管理接口，以便于基于 LSM-tree 的 KV 存储可以通过一组类似的常见文件操作（即 kvread/kvwrite/kvsync/kvunlink 操作），在基于 IMR 的 HDD 上通过有限的修改轻松访问 SSTable。KVIMR 由写处理器、读处理器、同步处理器、释放处理器以及一个缓冲区和一个 SSTable-to-Track 映射表组成，KVIMR 引擎维护了 SSTable-to-Track 映射，并为主机系统的 DRAM 空间分配了写缓冲区，并包含四个 handler 分别接管基于 LSM-tree 的 KV 存储的 kvread、kvwrite、kvsync 和 kvunlink 操作。其中 SSTable-to-Track 映射表维护了一个 SSTable 和为其分配的多个 IMR 磁道之间的一对多的关系，因为主流的基于 LSM-tree 的 KV 存储实现的 SSTable 大小通常大于 IMR 磁道大小。每当后台线程（例如压缩合并进程）调用 kvwrite 操作将 SSTable 数据传递到中间件，整个 SSTable 可以通过同步处理器一次性持久化到基于 IMR 的 HDD 中。一旦 SSTable 被持久化，KVIMR 立即释放相应的 DRAM 空间来保持对写缓冲区的低需求。同步处理器负责根据不

同的磁道分配策略将缓冲的SSTable持久化到 IMR 磁道，每当 KV 存储的后台线程需要调用 kvsync 来确保 SSTable 数据被持久化到 HDD。此外，在 SSTable 的持久化过程中，同步处理器还会执行 RMW 方法重写磁道，以确保写入数据的崩溃一致性，并在指定的 SSTable 的分配的磁道之间建立关系，以方便后续访问持久化的 SSTable。此外在运行时，为了有效地确认是否需要 RWM 方法，KVIMR 还在 DRAM 空间中维护一个位图，以跟踪磁道的分配状态。然而，这个位图不需要持久化到 HDD 中，因为它可以通过扫描映射表轻松重建。读处理器通过映射表从基于 IMR 的 HDD 的相应磁道中获取 SSTable 中被请求的部分。释放处理器负责从 HDD 中移除一个 SSTable，并将相应的磁道标记为空闲磁道，当 KV 存储的后台线程调用 unlink 操作时，通过设置映射表中相应的条目更新磁道状态。在这四个处理器中，同步处理器扮演着最关键的角色，因为 SSTable 如何被持久化到基于 IMR 的 HDD 的磁道中，可能会在很大程度上影响较为耗时的 RMW 操作的次数和后台压缩合并过程的效率。

在基于 LSM-tree 的 KV 存储中，不同级别的 SSTable 通常有不同的压缩合并频率。首先压缩合并过程通常以级联的方式从较小的层级到较大的层级进行；其次，不同级别的层大小限制随着级别的增加呈指数级增长。因此，较小级别的 SSTable 比较大级别的 SSTable 更容易被频繁的压缩合并，换句话说，较大级别的 SStable 的生命周期比较小级别的更长。除此之外，基于 LSM-tree 的 KV 存储在不同的压缩合并过程中存在一个非常特殊的局部访问性质，即压缩局部性。一轮压缩合并的 SSTable，更有可能会参与到下一轮的压缩合并过程。也就是说，存在一种特殊的局部倾向，即一轮压缩过程生成的 SSTable 很可能会参与到后一轮的压缩合并过程中。

基于上述的访问局部性和压缩局部性，KVIMR 提出了两种优化磁道分配算法，第一种算法为基于 Level 的双层磁道分配+松弛顺序的磁道分配算法。第二种算法为合并 RMW 的 SSTable 持久化算法。算法一的主要思想是如果底部磁道可以分配给层级更大的 SSTable，底部磁道将被寿命更长的 SSTable 占用，而不会被寿命更短的

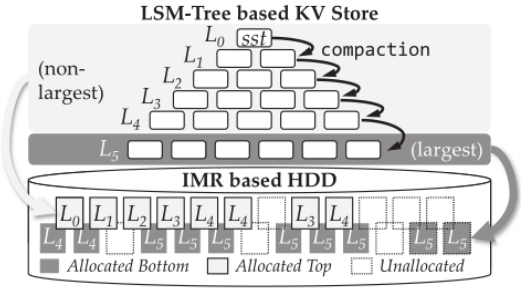

图 3 基于 LSM-tree 的 KV 存储磁道分配策略

其他 SSTable 频繁的重新分配。也就是说，第一步的关键思想是分配底部磁道以适应更大级别的 SSTable，从而使得分配底部磁道时产生的 RMW 的概率最小化。如图 3 所示，KVIMR 将“当前最大”级别（如 L5）的 SSTable 分配底部磁道，而非最大级别（如 L0 ~L4）的 SSTable 分配顶层磁道。这是因为，考虑到基于 IMR 的 HDD 底层磁道和顶层磁道的容量大致相同，而基于 LSM-tree 的 KV 存储各级的大小限制呈指数级增长，当 LSM-tree 增长时，底层磁道将由最大级别的 SSTable 完全分配。第二步确定哪个磁道在特定的层应该分配来容纳一个 SSTable。首先，根据压缩局部性，一轮压缩合并的 SSTable，更有可能会参与到下一轮的压缩合并过程。其次，由于基于 IMR 的 HDD 采用了与基于 CMR 的

HDD 类似的旋转磁盘机制，因此在基于 IMR 的HDD中顺序访问数据的效率也比随机访问数据的效率高。因此，KVIMR 尽可能顺序地容纳由一轮压缩过程生成的 SSTable，既可以顺序写入具有较高顺序写性能的磁道，也可以被后面几个具有较高顺序读性能的压缩合并过程顺序压缩。

为了进一步提高在RMW发生时将缓冲的SSTable 从写缓冲区持久化到 HDD 中,KVIMR 提出的第二个算法主要思想是将多个RMW操作重新排序为一个“合并的 RMW”，从而显著减少所需调用的的同步方法次数，并避免冗余磁道重写，同时仍然确保崩溃一致性。合并的RMW方法处理所有分配的底部磁道会导致顶部磁道的批量重写，首先，在调用同步函数前，KVIMR 将所有涉及的相邻顶层磁道的有效数据批量备份到备份缓冲区中，以确保后续把这些备份数据批量的写回到顶层磁道。KVIMR 会先将未写的 SSTable 以最高的优先级批量写入到相应的底层磁道。这是因为，如果 SSTable 中未写的部分先写顶层磁道，随后对相邻的底层磁道的写操作会导致额外的顶层磁道重写操作。最后，KVIMR 调用类似异步的函数，以确保对底层磁道的数据被持久化到HDD中。与第一种算法相比，算法二不仅显著减少了所需要的同步函数的调用次数，还避免了顶层磁道的冗余备份和回写（因为所涉及的顶层磁道只备份和回写一次）。此外，合并 RMW 方法很好的确保了 SSTable 的数据崩溃一致性。

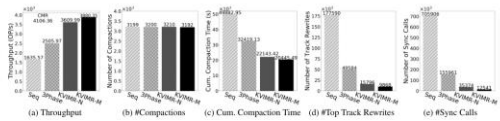

图 4 实验结果

由图 4 实验结果可以看出，KVIMR 提出的基于 Level 的双层磁道分配+松弛顺序的磁道分配算法和合并 RMW 的 SSTable 持久化算法相比传统的磁道分配算法均有着极大的性能提升。

4 总结与展望

基于 LSM-tree 的 KV 存储可以从存储器和磁道分配策略两个方向进行研究，而一些著名的 KV 存储也提出了自己的解决方案，如 SMRDB 提出将 SSTable 的大小扩大到基于 SRMR 的 HDD 的 band/zone 大小，以避免磁道重写；将 SSTable 排列成两个级别，并且 SSTable 的关键范围在同一级别内重叠。另一种名为 SEALDB 的研究建议将压缩过程中涉及的 sstable 分组为一个集合，然后顺序地将一组 SStable 写入到一个可变大小的动态带中，以缓解磁道重写问题。最近的一项名为 GearDB 研究，提出将相同级别的 SSTable 顺序写入相同的 band/zone，并进一步引入了 Gear compaction 以避免基于 SMR 的 HDD 中的垃圾收集。这些研究缓解了 SMR 的轨道重写问题,但盲目应用到基于 IMR 的 HDD 可能是无效的。介于文章篇幅和作者精力原因，本文重点总结了基于 IMR 的 HDD 存储器研究以及基于 LSM-tree 的 KV 存储磁道分配策略，以及将二者结合到一起共同作用所产生的提升。作者之后的工作将放在上述基于 SMR 的 HDD 上的优化算法作用到基于 IMR 的 HDD 上是否会带来性能上的提升的研究上。

参考文献

[1] Abutalib Aghayev, Mansour Shafaei, and Peter Desnoy- ers. Skylight—a window on shingled disk operation. ACM Transactions on Storage (TOS), 11(4):16, 2015.

[2] Ahmed Amer, JoAnne Holliday, Darrell DE Long,Ethan L Miller, Jehan-François Pâris, and Thomas Schwarz. Data management and layout for shingled magnetic recording.IEEE Transactions on Magnetics,47(10):3691–3697, 2011.

[3] Sugimoto, Cassidy R., Hamid R. Ekbia, and Michael Mattioli, eds. Big data is not a monolith. MIT Press, 2016.

[4] Li Y, Liu Z, Lee P P C, et al. Differentiated Key-Value Storage Management for Balanced I/O Performance[C]//2021 USENIX Annual Technical Conference (USENIX ATC'21). USENIX Association. 2021.

[5] Kaizhong Gao, Wenzhong Zhu, and Edward Gage. Interlaced magnetic recording, August 8 2017. US Patent 9,728,206.

[6] Kaizhong Gao, Wenzhong Zhu, and Edward Gage. Write management for interlaced magnetic recording devices, November 29 2016. US Patent 9,508,362.

[7] Steven Granz, Jason Jury, Chris Rea, Ganping Ju, Jan-Ulrich Thiele, Tim Rausch, and Edward Gage. A real density comparison between conventional, shingled, and interlaced heat-assisted magnetic recording with multiple sensor magnetic recording. IEEE Transactions on Magnetics, PP:1–3, 09 2018.

[8] Patrick O'Neil, Edward Cheng, Dieter Gawlick, and Eliz-abeth O'Neil. The log-structured merge-tree (lsm-tree).Acta Informatica, 33(4):351–385, 1996.

[9] Raju P, Kadekodi R, Chidambaram V, et al. Pebblesdb: Building key-value stores using fragmented log-structured merge trees[C] roceedings of the 26th Symposium on Operating Systems Principles. 2017: 497-514.

[10] Fenggang Wu, Bingzhe Li, Baoquan Zhang, Zhichao Cao, Jim Diehl, Hao Wen, and David HC Du. Track lace: Data management for interlaced magnetic recording. IEEE Transactions on Computers, 2020.