Larger, Cheaper, but Faster: SSD-SMR Hybrid Storage Boosted by a New SMR-oriented Cache Framework

孙考毅

摘要

大数据时代下，各类数据（例如科学计算领域气象学与天文学）以极快的速度增长，这对磁盘的存储能力与成本提出了巨大挑战。通过使用新的叠瓦式磁记录 (SMR) 技术，新兴的 SMR 磁盘实现了更高的存储密度与更低的成本。基于 Flash 的 SSD 与 SMR 磁盘可以用来构建一种新型混合存储架构 (SSD-SMR)，基于此架构的磁盘比广泛使用的传统磁盘有着更高的存储密度，同时还能保持成本不变甚至降低。然而，SSD-SMR 混合存储架构的缺点在于其弱细粒度中的写放大导致的随机写性能降低问题，当读写请求随机分布在整个磁盘的 LBA (Logical Block Address) 范围时，该问题异常严重。传统的缓存算法没有考虑待更新数据与磁盘待写入区域的分布关系，从而导致较差的读写性能。本文复现的论文中，提出了一种新的面向 SSD-SMR 的缓存架构，Partially Open Region for Eviction (PORE)。该缓存架构通过 LBA 分布范围对待更新数据进行写入限制，以达到混合存储架构下缓存使用效率与 SMR 写放大问题之间的平衡。

关键词：混合存储架构；SSD-SMR；写放大

1 引言

大数据时代已经到来，全球数据总量庞大，越来越多的数据被用于大数据应用中的各种数据分析，如科学计算 (例如，在气象、天文学或生物学领域)、互联网网页和用户日志、医疗图像等。不仅如此，这些数据还在以惊人的速度增长。快速增长的数据对大数据存储系统的可拓展性、成本等各方面都提出了很高的要求。一种新型的存储技术——叠瓦磁记录式磁盘（SMR），SMR 通常与 Flash-based SSD 来构建一个新的混合存储系统。该存储系统相对传统的存储系统，有更高的存储密度及更低的成本。

2 动机

与传统磁盘相比，SMR 在处理小的随机写操作时存在性能挑战，因为对于小的随机写操作，SMR 存在写放大问题，这是其高存储密度的副作用。现有的技术在文件系统下面的块层中部署 Flash-based SSD 作为缓存，以增强磁盘的性能。但由于 SMR 的特性，缓存方案将遇到新的挑战，当写数据的逻辑地址范围从 56GB 增加到 1TB 时，由于 SMR 写放大率的增大，其随机写性能迅速下降了。传统的缓存替换算法 (如 LRU、LIRS、ARC 等) 在选择驱逐对象时只考虑数据块的局部性，导致 SMR 写放大严重。

为了解决上述问题，本次进行复现的论文提出了一种新的面向 SMR 的缓存框架，即部分开放区域驱逐 (partial Open Region for Eviction, PORE)。它适当地限制了 Flash-based SSD 缓存中驱逐数据的逻辑地址范围，通过更好地权衡缓存数据的局部性和 SMR 写放大之间的关系来提高混合存储的整体性能。

3 相关工作

3.1 SMR 设备的优化

构建 SMR 设备的主要挑战在于，根据原始的片状磁记录设计，在执行随机写入时，重叠轨道中的内容会被覆盖。Y. Cassuto 等人[1]为 SMR 设备提出了一个新的指示系统的概念。他们提出了一个名为 Shingled Block（S-block）的存储单元，它与 SMR 带的概念相似，其间是物理层面的安全间隙。S-块在一个循环缓冲器中管理；新写入数据的逻辑地址被映射到循环缓冲器的头部。通过地址映射方案，这个指示系统可以支持随机写入。垃圾收集算法通过将有效块移到头部来回收循环缓冲区尾部的 S 块。基于指示系统，D. Hall[2]等人提出了一个方案，将一个 E 区从遮蔽区中划分出来作为缓存缓冲区，以加速随机写入。

为了提高垃圾收集的效率，D. Park 等人[3]带来了一种新颖的设计，将数据分为热的和冷的，这使得过时的块更集中在热带中。S. Jones 等人[4]提出了一种带状压缩算法，以减少带状压缩过程中的数据移动，该算法不仅关注压缩最空的带状，还考虑了带状中数据的写入频率。在地址映射领域，W. He 等人[5]提出了一种新颖的映射方法，先将数据定位在互不重叠的轨道中，牺牲空间效率来获得更高的性能。此外，Ma 等人[6]提出使用基于闪存的固态硬盘来取代位于轨道上的持久性写缓冲区，并提出了一种基于带状分布知识的缓存算法来管理写缓冲区。

3.2 基于 SMR 的系统的优化

另一个研究方向是基于主机管理的 SMR 磁盘，在上层文件系统层做转接工作。C. Jin 等人[7]设计了一个新的文件系统，称为 HiSMRfs，可以在没有 SMR 产品的 SMR 转换层（STL）的情况下支持随机写入。G. Gibson 等人[8]提出了一个新的 SMRfs，它基于由固态硬盘和 SMR 磁盘组成的混合系统工作，并单独管理元数据以提高性能。S. Kadekodi 等人[9]提出了一个名为 Caveat-Scriptor 的接口方案，支持基于静态映射的任意写入。针对特定用途的 SMR 文件系统也被提出，例如用于视频服务器的 SFS 和大数据应用。

除了文件系统，也有一些关于基于 SMR 磁盘的数据库的研究。例如，P. Rekha 等人[10]提出了一个用于 SMR 磁盘的键值数据存储，利用其更好的连续写入性能。

此外，A. Aghayev 等人[11]提出了一种方法来逆转驱动器管理的 SMR 驱动器的关键属性，包括关于持久性写缓冲区的细节，带状大小等，这对 SMR 相关的系统优化有很大帮助。

3.3 混合存储中的 SSD 缓存优化

基于闪存的 SSD 由于其高性能和高存储密度而被广泛用于企业存储系统。由固态硬盘和磁盘组成的混合存储系统也是现代数据中心的重要存储形式，因为它在成本性能和数据可靠性方面具有优势。更重要的是，有许多工业产品或开源系统来优化混合存储中的 SSD 缓存，如 Facebook Flashcache, Linux dm-cache, EMC FAST cache, 以及 Exadata 数据库机中的 Oracle Smart Flash Cache。

除了一些传统的通用缓存算法，如 LRU、MQ、LIRS、ARC 等，还有一些专门为基于 SSD 的缓存优化的缓存算法。例如，Solaris ZFS 文件系统中的二级 ARC（L2ARC）技术，SieveStore 和 LARC 增加了不同类型的新数据过滤器来提高缓存数据的质量，同时减少缓存的更新频率。WEC 和 ETD-Cache 通过适当地延长它们在基于 SSD 的缓存中的停留时间来保护流行的缓存数据，以避免过早地被驱逐。RIPQ、Pannier 和 SRC 将小块数据打包成大容器，并以大容器为单位替换缓存数据。由于闪存芯片中

的写入放大率降低，这种机制可以提高 SSD 的 I/O 性能。

Skylight[11]是一种将缓存通过静态映射分配给磁盘区域的映射方法。该方法的有点在于将物理相关性高的数据集中映射在某几个缓存块中，当缓存块中的数据需要写入磁盘时，集中的数据能够最大限度地增加磁盘更新操作的带来的收益，这在一定的程度上减少了写放大带来的负面影响。

此外，M. Canim 等人[12]为数据库系统中 RAM 和硬盘之间的 SSD 缓存层提出了一种缓存算法。该算法监控磁盘访问模式，以识别磁盘的热点区域。磁盘热区的区块有更高的优先级被缓存在 SSD 中，以避免冷区的缓存污染。

总的来说，尽管已经有一些专门为基于 SSD 的缓存设计的优化方案，面向 SMR 的 SSD 缓存方案仍然值得探索。

4 本文方法

4.1 本文方法概述

为了减少 SMR 的写入放大率，本文将 SMR LBA 范围的某些部分设置为 Open Region，其余部分为 Forbidden Region。如图 1所示，干净的缓存块不受 Open Region 和 Forbidden Region 设置的影响；对于脏的缓存块，只有位于 Open Region 的（如图 1中的 a、b、c 块）才允许被驱逐；那些位于 Forbidden Region 的块在选择受害者时将被跳过（如 d 块）。

在这种情况下，SMR 的写入放大率可以降低。例如，当开放区域覆盖 200GB 空间，而 SMR 磁盘的持久性写缓冲区是 20GB，这种情况下的写放大率大约是 10（即 200GB/20GB）。如果没有开放区域的设置，写缓冲区中的块可能位于任何区段，所以我们可能需要重新写整个 SMR 磁盘来写回它们。这导致了更大的写入放大率（例如，5 TB / 20 GB = 256）。

如图 1所示，这种机制被称为部分开放区域驱逐（PORE），为脏块的驱逐开放部分 SMR 磁盘空间，以减少 SMR 写放大，并定期选择 SMR LBA 范围的适当部分作为开放区域，以保持良好的缓存命中率。通过结合 LBA 感知的驱逐限制和块级缓存替换方式（而不是一起驱逐位于大磁盘区域的所有缓存块），PORE 可以通过实现低 SMR 写放大率和高缓存命中率来改善混合存储的性能。

4.2 工作流程

如图 1所示，PORE 不是一个特定的缓存替换方案，而是一个开放的缓存框架。任何其他的缓存替换算法都可以在 PORE 中使用，以管理 SSD 缓存中的缓存块。PORE 包括两个核心模块，即基于区域的驱逐和周期性区域划分。前者负责调用部署的缓存替换方案（例如 LRU）来驱逐其中一个缓存块，而后者则定期更新开放区域和禁止区域的划分。

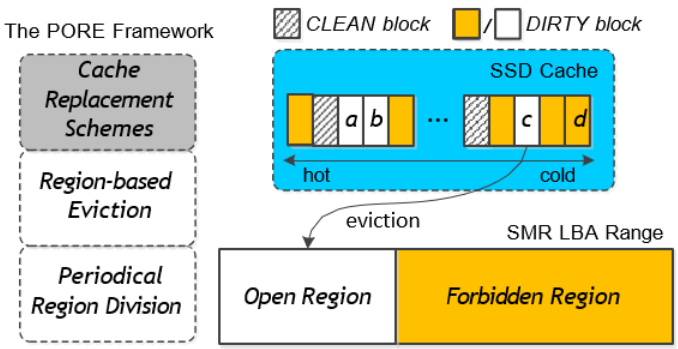

图 1: 系统架构图

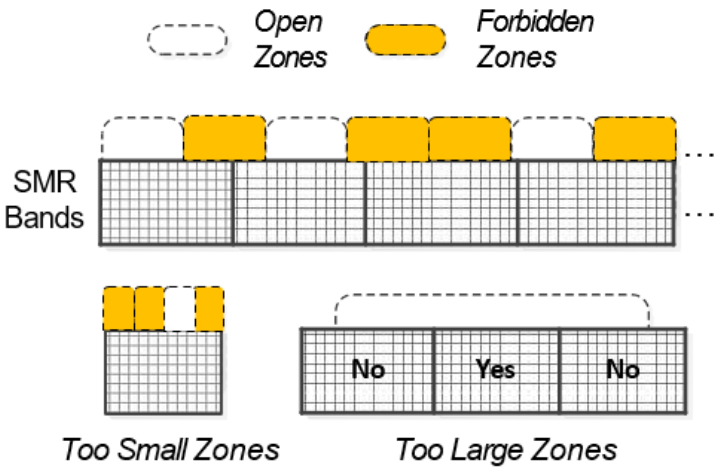

图 2: 开放区域的基本单位

4.2.1 基于区域感知的驱逐机制

在 PORE 中，作者在缓存的脏块的元数据中增加了一个新的标签，以识别该块所属的区域（即，开放区域或禁止区域），每块消耗 1 比特，这不会给内存空间带来明显的负担。当需要进行驱逐时，将根据部署的缓存方案选择最冷的干净块或开放区域的脏块。作者仍然采用块级的缓存替换，而不是把块放在一起驱逐，因为这样可以保护热块不被驱逐以保持高的缓存命中率。

然而，这可能会引入额外的时间来寻找一个受害者区块，因为它可能不是缓存队列中的最后一个。以图 1为例，如果采用 LRU，没有 PORE 的原始缓存替换可以立即从缓存块阵列或链接列表中获得最冷的块（即 d）。当它在 PORE 中工作时，必须要找到受害者，直到第三个（即 c）。然而，所有这些操作都发生在内存中，在 I/O 密集型的存储系统中不会对整体性能有太大影响。更重要的是，对于像 LRU 这样基于队列或队列的缓存算法，作者为 Open Region 中的所有缓存块引入了一个额外的链接列表，以提高查找过程。

4.2.2 基本区域区分单位

当设置开放区域时，最好的计划是将其与 SMR 带的边界对齐。然而，SMR 磁盘中的带子大小不确定（例如，希捷 5T SMR 磁盘 [10] 在 17□36MB 范围内有不同的带子大小），但是作者不知道驱动器管理的 SMR 产品的确切带子大小分布。因此，他们首先将整个 SMR 磁盘地址空间划分为具有固定大小的区域，并采用区域作为基本单位，形成 Open Region 和 Forbidden Region，如图 2。在开放区中的区域被称为开放区，而在禁止区中的区域被称为禁止区。

过小的分区不是一个好的选择，因为它将导致大量的分区，带来管理它们的高额开销。更重要的是，过小的区段尺寸设置会增加一个区段和其相关的开放区段之间出现大尺寸差异的风险，就像图 2所示的情况。因此，当开放区与带状区相比非常小的时候，驱逐这个开放区的块会导致很大的写放大率。另一方面，过大的区域设置也不利于性能的提高，因为它可能会把不同类型的带子合并到一个大的区域。其中一些频段可能适合作为一个开放的区域，而另一些则不适合。例如，一个包含许多热门脏区的频段本身就不应该被选为开放区，但当其相关大区的平均人气被其相邻的频段降低时，它就可能被卷入开放区。因此，作者为大区设定一个适度的规模，一般与 SMR 带的最小规模在同一数量级上。

4.2.3 定期开放区域驱逐

假设区域划分是固定的，作者在开放区域驱逐一个热块，而不是在禁止区域驱逐一个冷块，从而降低了缓存的命中率。因此，在 PORE 中，将定期改变这两种区域的设置。在这种情况下，位于不同磁盘区域的冷脏块都有机会被轮流驱逐。在每个区域划分时期的开始，系统将根据上一时期所有区域的访问记录，选择一些区域组成开放区域，而其他区域组成禁止区域。特别是，第一次区域划分发生在 SSD 缓存满载的时候。

当写进 SSD 缓存的脏块数量达到指定的阈值（即周期长度）时，新的周期开始设置新的开放区域和禁止区域。SMR 磁盘的写入放大率可以被估计为公式 2，其中 Sor 是 Open Region 的大小，Swb 是 SMR 写入缓冲区的大小，Lp 是 PORE 的区域划分周期的长度。写入放大率远远低于 Swhole disk=Swb 的原始值。

只有当位于同一 SMR 带的脏块已经在 SMR 写缓冲区时（例如图 3（a）中的 a1、a2 和 a3 块），它们才能同时被冲到 SMR 带以减少写放大。当周期长度长于写缓冲区大小时，尚未从 SSD 中驱逐的缓存块（例如，块 a4）不能与 a1、a2 和 a3 一起被冲到 SMR 带。因此，写入放大率等于 Sor=Swb，分母的值是恒定的，但是分子表示的是相应的开放区域的大小，必须被扩大以在下一个时期提供足够的驱逐受害闪存块，导致更大的写入放大。

相反，当周期长度小于写缓冲区的大小时，来自不同轮开放区域的块可能在写回带子之前混合在 SMR 驱动器的持久性写缓冲区中。如图 3（b）所示，由于周期 B 与周期 A 有不同的开放区域，周期 B 的块通常不能与周期 A 的块一起冲到 SMR 带中。一般来说，开放区域的大小随着周期长度的增加而增加，当周期长度不长于写缓冲区的大小时，Rwa 的值不会有太大变化。

然而，由于 PORE 的区段与 SMR 的区段并不完全一致，一个开放区域的覆盖区段的总大小通常比所有开放区域的总大小大一些。当周期长度太小时，波段和区域之间的尺寸差距将是不可忽略的，从而导致写入放大率稍大。此外，太小的周期长度也会增加选择开放区的计算开销，所以周期长度的最佳设置正好是 SMR 写缓冲区的大小。

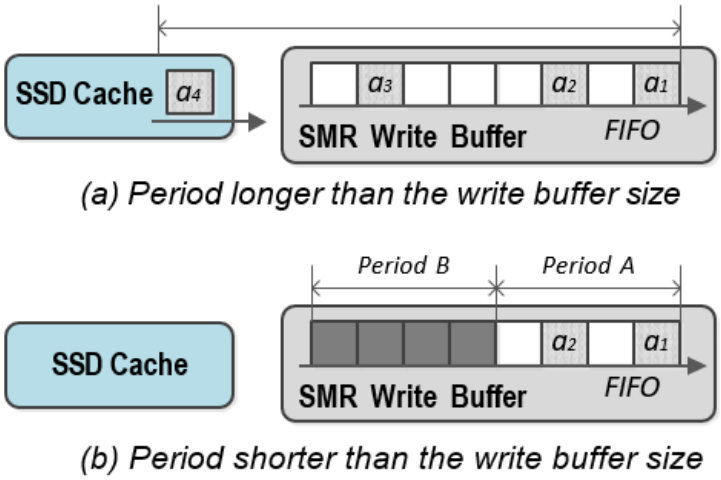

图 3: 设置不同的周期长度

5 复现细节

5.1 与已有开源代码对比

本论文没有开放源代码，复现过程中没有参考任何相关源代码。

5.2 主要实现的内容

5.2.1 SMR 仿真及其接口定义

SMR 作为该系统中对底层的核心硬件，承担着重要的读写请求任务。SMR 的存储单位是 sector，任意时候都可以对特定 sector 进行读取操作，但 sector 的写入操作有一定限制。对于特定状态的 sector（写入后不影响其他存储单元的 sector），可以通过追加写的形式进行写入操作。其余的 sector 则需要通过更新其所在的整个带状区域来完成其数据更新，即 SMR 的 RMW 操作，该操作会占用大量磁盘资源并使磁盘短时间内无法提供服务。

Procedure 1 SMR 的基本操作.

Input: LSN *l*

Output: datas *res*

$res = ReadSector(LSN)$

$return(res)$

Input: LSN *l,* data *d*

Output: success *res*

$AppendSector(LSN, data)$

$return(res)$

Input: LSN *l,* data *d*

Output: success *res*

$temp = ReadBand(LSNToLBN(LSN))$

$temp = Modify(temp, data)$

$res = WriteBand(temp)$

$return(res)$

5.2.2 Flash 仿真及其接口定义

Flash 的基本读写单位是闪存页，基本的擦除单位是闪存块，闪存块在混合存储架构中承担了主机到 SMR 磁盘间的缓冲区，减轻了磁盘的读写压力，对混合存储架构整体性能的提升起着重要作用。

Procedure 2 Flash 的基本操作.

Input: PageAdd *p*

Output: datas *res*

$res = ReadPage(p)$

$return(res)$

Input: PageAdd *p,* data *d*

Output: success *res*

$WritePage(p, data)$

$return(res)$

Input: Block *b*

Output: success *res*

$CheckValidPage(b)$

$res = EraseBlock(b)$

$return(res)$

5.2.3 EF-SMR 读写仿真及其接口定义

混合磁盘需要向主机提供正常的读写服务，发挥着数据持久化的重要作用，是计算机系统总的重要一环。

Procedure 3 EF-SMR 的基本操作.

Input: LSN *p*

Output: datas *res*

$ifFindCache(LSN)$

$res = ReadCache(LSN)$

$else$

$res = ReadSMR(LSN)$

$endif$

$return(res)$

Input: LSN *l,* data *d*

Output: success *res*

$ifWriteCache(p, data)$

$res = true$

$else$

$Reclaim()$

$flag = TryWriteCache(LSN)$

$iff lag == false$

$WriteSMR(LSN)$

$return(res)$

5.2.4 SSD-SMR 缓存回收策略及其表示

缓存架构适时对缓存块进行更新，依据系统的各项参数，调整下一次的部分开放区域。

Procedure 4 基于区域部分开放的缓存回收策略.

Input: Signal s

Output: success *res*

$GetOpenRegion()$

$data = ReadOpenRegionData()$

$res = SMRRMW(data)$

$UpdateOpenRegionSetting(s)$

$return(res)$

5.3 实验环境搭建

创建 C++ 工程项目，在其中进行硬件的仿真，采用的参数与需要进行对比的 Skylight 中的参数一致，磁盘转速 7200rpm，闪存读、写、擦除时间分别为 0.2ms、1.3ms、1.5ms。

6 实验结果分析

对平均相应时间、尾延迟、触发回收策略的次数三个指标进行实验并做归一化分析，得到下面的数据。

6.1 平均响应时间

将平均相应时间进行对比，如图 4所示。在进行测试的数据集中，PORE 比 Skylight 的平均相应时间均有减少，说明磁盘服务性能均有提升。PORE 的响应时间平均比 Skylight 减少 50.56%，最大减少 79.30%。因为部分开放驱逐区域使得 SMR 磁盘在一定程度上减少了 RMW 操作，使得整体性能提升。

6.2 尾延迟

将尾延迟进行对比，如图 5所示。在进行测试的数据集中，PORE 比 Skylight 的尾延迟均有减少，说明磁盘服务性能均有提升。PORE 的尾延迟平均比 Skylight 减少 15.02%，最大减少 18.60%。因为部分开放驱逐区域使主机进行读写操作时出现极端情况的概率降低，使得整体性能提升。

6.3 平均迁移时间

将平均迁移时间进行对比，如图 6所示。在进行测试的数据集中，PORE 比 Skylight 的尾延迟均有减少，说明磁盘服务性能均有提升。PORE 的迁移时间平均比 Skylight 减少 15.47%，最大减少 18.62%。因为部分开放驱逐区域使主机在进行数据迁移操作时，需要操作的数据量减少了，使得整体性能提升。

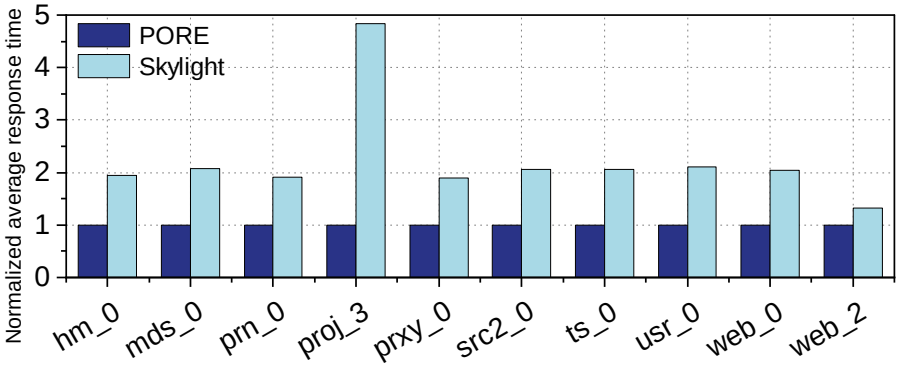

图 4: 归一化平均响应时间

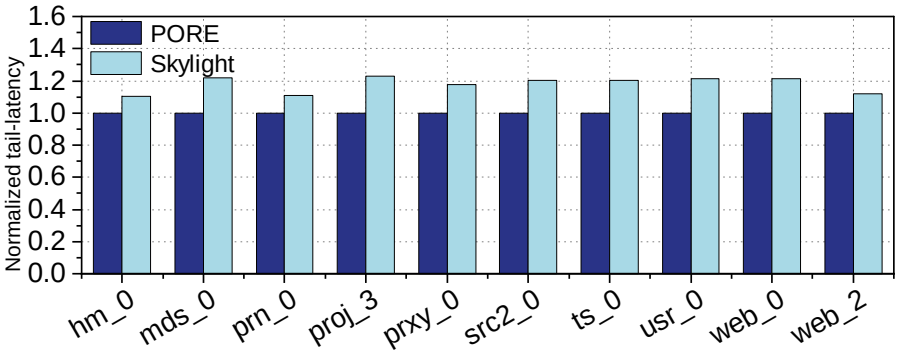

图 5: 归一化最大尾延迟

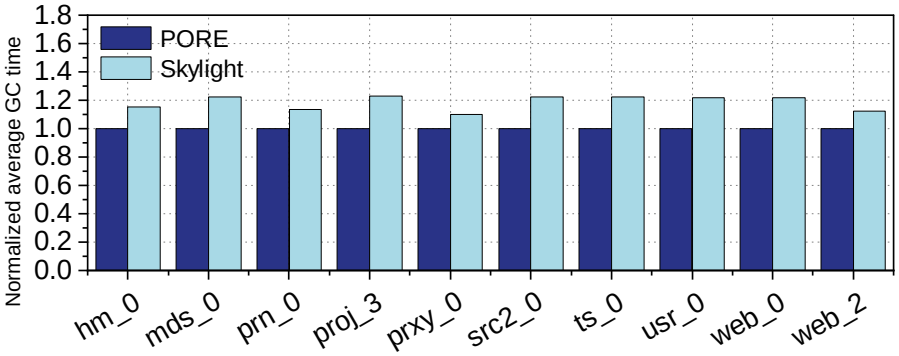

图 6: 归一化平均迁移时间

7 总结与展望

本报告针对所选文章进行复现，归纳分析了该文章技术的应用背景、技术要点；对其中的主要技术进行了无源码复现；将复现后的程序与相关工作进行比较，得到了较好的结果。本次实现过程中，存在一些不足。由于缺乏源码，导致在具体技术实现时难免有所疏漏，对于不理解的难点有所省略。在后续的学习研究中，我会进一步研究文章中尚未完全理解的难点，丰富实验内容。

参考文献

[1] HALL D, MARCOS J H, COKER J D. Indirection systems for shingled-recording disk drives[C]//In Mass Storage Systems and Technologies (MSST). 2010: 1-14.

[2] HALL D. Data handling algorithms for autonomous shingled magnetic recording hdds[J]. Transactions on Magnetics, 2012, 48(05): 1777-1781.

[3] PARK D, LIN C I, DU D H. H-swd: A novel shingled write disk scheme based on hot and cold data identification[C]//In10th USENIX Conference on File and Storage Technologies (FAST12). 2012: 1-12.

[4] JONES S N, AME A, MILLER E L, et al. Data handling algorithms for autonomous shingled magnetic recording hdds[J]. ACM Transactions on Storage (TOS), 2016, 12(1): 1-14.

[5] HE W, DU D H. Novel address mappings for shingled write disks[C]//HotStorage. 2014: 1-8.

[6] MA L, XIAO W, DONG H, et al. Most: A high performance hybrid shingled write disk system[C]//In

Proceedings of the 22nd National Conference of Information Storage. 2016: 1-6.

[7] JIN C, XI W Y, CHING Z Y, et al. Hismrfs: A high performance file system for shingled storage array [C]//In Mass Storage Systems and Technologies (MSST). 2014: 1-6.

[8] GIBSON G, POLTE M. Directions for shingled-write and twodimensional magnetic recording system architectures: Synergies with solid-state disks[C]//Parallel Data Lab, Carnegie Mellon Univ. 2009: 1-14.

[9] PIMPALE S K S, GIBSON G A. Caveatscriptor: Write anywhere shingled disks[C]//In HotStorage. 2015: 1-12.

[10] HUGHES R P J, MILLER E L. Smrdb: keyvalue data store for shingled magnetic recording disks[C]// In Proceedings of the 8th ACM International Systems and Storage Conference. 2015: 1-18.

[11] AGHAYEV A, SHAFAEI M, DESNOYERS P. Skylighta window on shingled disk operation[J]. ACM Transactions on Storage (TOS), 2015, 11(4): 1-16.

[12] CANIM M, MIHAILA G A, BHATTACHARJEE B, et al. Ssd bufferpool extensions for database systems[J]. Proceedings of the VLDB Endowment, 2010, 3(1-2): 1435-1446.