doi:10.3772/j.issn.1002-0470.2017.02.005

高性能混合结构的瓦记录磁盘系统的设计与实现①

马留英②*** 肖文健*** 董欢庆* 许 鲁*

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院大学 北京 100049)

摘 要 对新的磁记录技术——瓦记录(SMR)进行了性能分析,针对 SMR 叠瓦式的磁道布局方式严重限制了其随机写性能,希捷公司的瓦记录磁盘(SSWD)在磁盘内部使用持久缓存虽能解决 SMR 叠瓦特性导致随机写访问受限,但其在持续随机写入的情景下性能表现很差的问题,设计了一种结合固态硬盘(SSD)的混合瓦记录磁盘(HSWD)结构;使用 SSD 作为持久缓存,实现了全相联和组相联两种持久缓存到本地存储的映射策略,以及三种持久缓存回收策略——LRU、FIFO 和 MOST,并通过试验和理论分析分别对比了两种映射策略和三种回收策略。最后,通过测试对比 HSWD、SSWD 和 Flashcache 的性能,结果表明,HSWD 较 SSWD 和 Flashcache 在持续随机写性能上有了明显提升,例如在回放 hm0 的测试中,HSWD 的性能较 SSWD 提升 2.25 倍,较 Flashcache 提升 1.32 倍。

关键词 瓦记录(SMR),瓦记录磁盘(SSWD),固态硬盘(SSD),混合系统,缓存策略,回收策略

0 引 言

一直以来,硬盘驱动器(hard disk drive, HDD)因具有容量大、价格低、性能稳定的特点而成为最主流的存储介质,随着信息爆炸和大数据时代的到来,对存储介质的容量提出了更高的要求,其相对于其它存储介质的优势将更加明显。然而,受限于超顺磁效应,使用垂直磁记录技术的 HDD,其存储密度不可能达到 1T/in^2,因此,需要使用新的磁记录技术[1]。于是,出现了一些新的技术,包括:比特晶格磁记录(bit patterned magnetic recording, BPMR)[2],热辅助磁记录(heat assisted magnetic recording, HAMR)[3],微波辅助磁记录(microwave assisted magnetic recording, MAMR)[4],瓦记录(shingled magnetic recording, SMR)[5]等。前三种技术需要对底层介质或磁头进行大幅改变,并需要长时间巨大成本的投入,现阶段很难实现。瓦记录技术因其仅需对现有磁盘的物理结构和介质改变较小而成为第一个应用到市场的技术,希捷公司现已推出使用瓦记录技术的磁盘产品瓦记录磁盘(shingled write disk, SWD)[6]。

瓦记录的基本原理是相邻磁道就像屋顶上的瓦片一样重叠,缩小磁道间距,从本质上提升存储密度。瓦记录技术不影响读操作,但由于其采用叠瓦式的磁道布局方式,使其仅支持顺序追加写操作,随机写操作将严重受限[7-9]。希捷公司的瓦记录磁盘(seagate SWD, SSWD)通过将其内部的部分区域用作持久缓存(persistent cache, PCache)[10]以解决其随机写访问限制问题,并向后兼容现有的存储系统。然而,SSWD 在持续随机写入情况下性能表现很差。本文从磁盘内部入手,基于 Gibson 提出的瓦记录磁

① 863 计划(2013AA013205),中国科学院先导专项基金(XDA06010401)和中国科学院重点部署基金(KGZD-EW-103-5(7))资助项目。

② 女,1986 年生,博士生;研究方向:瓦记录存储等;联系人,E-mail: maliuying@ict.ac.cn

(收稿日期:2016-11-08)

盘(SWD)与固态硬盘(solid state drive,SSD)结合的混合存储系统的思想[11],使用 SSD 作为持久(persistent)缓存区域(PCache),并设计相应的策略,在块设备驱动层模拟实现了新结构下的混合瓦记录磁盘(hybrid SWD, HSWD)。本研究的结果既可直接用于主机管理类型的瓦记录磁盘的优化,亦可给瓦记录磁盘生产厂商提供有效的参考。本研究主要提出了一种混合的存储架构以解决瓦记录磁盘持续随机写性能低下的问题,并为该架构中的持久缓存设计了合理的数据管理方式,以高效地处理持续的随机写请求;实现了两种持久缓存映射策略和三种回收策略,并通过理论分析和实际测试对上述策略进行对比,结果表明,全相联映射策略更适合于瓦记录磁盘,而含缓存块最多(MOST)的回收策略在多数应用场景下能够提供更好的性能;将 HSWD 原型系统实现为一个 Linux 内核模块,并对该原型系统做了性能测试,测试结果表明,HSWD 原型系统的性能明显优于 SSWD 和 Flashcache。

1 背景和相关工作

瓦记录技术采取叠瓦式的磁道布局方式,其示意图如图 1 所示,磁道 N 与 N+1、N+2 重叠,当对磁道 N 上的数据进行修改时,磁道 N+1、N+2 上的数据会被破坏,因此需要先将磁道N-N+2的数据读出,修改后再写回,这个过程称为读-修改-写(read-modify-write, RMW)。为防止 RMW 扩散到全盘,引入了带(Band)的概念,带由一系列相邻的磁道组成,带间存在隔离区(Guard)以保证上一个带的最后一个磁道的数据写入不会破坏下一个带的第一个磁道上的原有数据,这样就将 RMW 操作限定在带内。

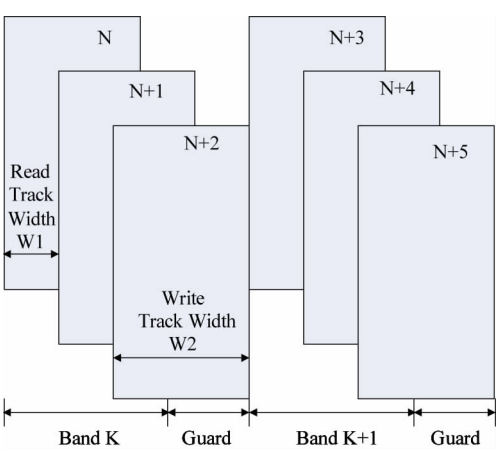

图 1 SMR 磁道布局

目前,解决瓦记录磁盘的访问限制问题的研究工作主要集中在两方面。一方面是设计新的磁盘数据管理方式。Cassuto 等[12]提出了两种间接系统:第一种采用静态映射的方式,使用组相联的方式来缓存 IO 请求并通过 RMW 进行回收缓存;第二种是基于环形日志结构的思想提出的 S-Block 的数据组织方式。Amer 等[13]提出了将 SWD 划分成日志访问区(log access zone, LAZ)和随机访问区(random access zone, RAZ),日志访问区用日志结构来管理,随机访问区用于存储元数据和缓存用户数据。Lin 等[14]将热数据识别技术引入 SWD 中来提升垃圾回收操作的性能。Luo 等[15]提出了一种基于段的数据布局和一种波形叠瓦的瓦记录磁盘系统,提升了瓦记录磁盘的性能和容量。另一方面是研究适合瓦记录磁盘的文件系统,如 ShingledFS[16], SFS[17], HiSMRfs[18]等。

另外,Aghayev 等通过逆向工程方式探测了 SSWD 内部结构[10]:由持久缓存区域(PCache)和本地存储区域(native region, NRegion)组成,如图 2 所示。其中,PCache 位于盘片的最外圈,并以日志结构的方式组织数据。NRegion 位于内圈,整个盘面按带(大小介于 15MB ~ 40MB)组织划分,带与带之间相互隔离。其写数据流程如图 3 所示,写请求首先会进入磁盘的内存缓存(memory buffer),如果凑齐整带的数据,则直接将整带数据顺序写入 NRegion;否则将数据写入 PCache,待 PCache 进行回收时通过 RMW 的方式将数据写回到 NRegion。PCache 采用日志结构[9]管理数据,存在如下问题:

(1) 采用日志结构管理 PCache,会引入大量的元数据。一方面是管理复杂,另一方面是元数据的读写访问导致磁头抖动从而影响数据写性能。

(2) PCache 单一的回收策略导致 RMW 执行效率降低。由于 PCache 采用了日志结构的管理方式,回收时必须先回收头部空间,使其必须采用先进先

出(FIFO)的回收策略[9],如图 4 所示。相同标记的块表示位于同一个带内,采用 FIFO 的方式回收时,会先回收第 K 个 Band 的数据,虽然回收了 3 个块,由于 SMR 的叠瓦特性实际上可用的块仅为在头部的 1 个块,需要继续执行 RMW 操作来回收块,当缓存压力较大时,这将导致较大的 IO 延迟。

(3) PCache 作为磁盘的一部分会引入较大的磁头抖动。比如对于读写混合类应用,当写请求发往 PCache 和读请求发往 NRegion 时,将会导致磁头的频繁抖动,同时在执行 RMW 操作时磁头的抖动也会影响 RMW 的效率。

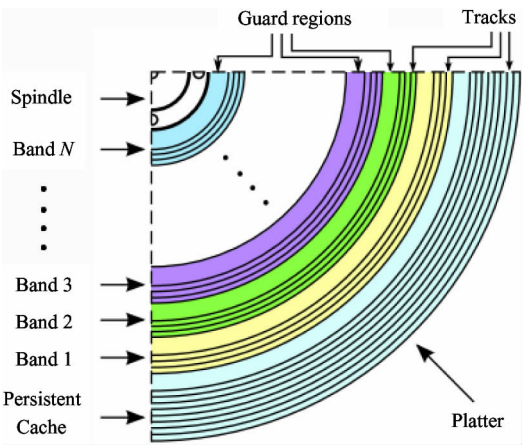

图 2　SSWD 的盘片结构[9]

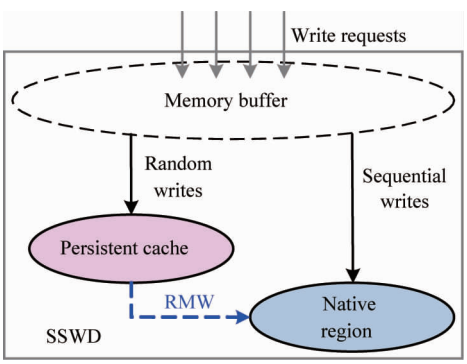

图 3　SSWD 的写请求处理流程(使用写缓存)

图 4　SSWD 的 PCache 回收策略

2　HSWD 系统结构

2.1　系统结构概述

图 5 是混合瓦记录磁盘(HSWD)的系统结构图。使用 SSD 来替代磁盘上部分区域作为持久缓存(PCache)区域有如下几个优势:(1) SSD 随机读写性能高,无需使用日志结构来管理,解决了随机写性能差,日志结构管理结构复杂且产生大量元数据的问题;(2) 使用了 SSD 的 PCache 可以随机写入,使得 PCache 可以采用更加灵活的回收策略,进而提高 PCache 回收效率和空间利用率;(3) 将 PCache 和本地存储区域(NRegion)独立,减少了磁头抖动,提高了读写性能和读-修改-写(RMW)的效率。

由于希捷公司的瓦记录磁盘(SSWD)的 NRegion 物理介质和磁头工作机制与 HDD 相同,因此在 HSWD 中使用 HDD 模拟 NRegion,为保证 HSWD 和 SSWD NRegion 的结构一致性,必须关闭 HDD 的写缓存。

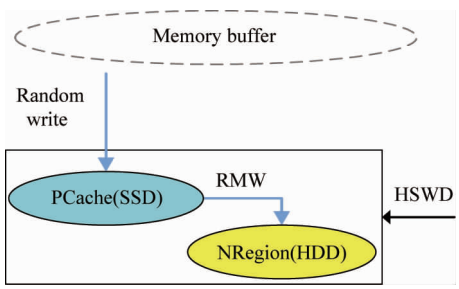

图 5　HSWD 系统结构

2.2　PCache 和 NRegion 之间映射策略

首先要确定的是 PCache 和 NRegion 之间的映射关系。传统的映射方式有两种:组相联和全相联。

组相联(Set-Associative)将 PCache 和 NRegion 划分成大小相同的组,组间采用直接映射,即 NRegion 的某个组只能映射到 PCache 中唯一的组,但是组内采取全相联映射,如图 6(a)所示。结合瓦记录磁盘中带的概念,将组大小设置为带大小,NRegion 有 n 个带,而 PCache 被划分成了 $m = 2$ 个组,分别为 C_0 和 C_1。第 i 个带会被映射到 PCache 中第 $i\%2$ 个组。Cassuto 等[8]提出的第一种间接系统就是使

用的组相联映射方式。

全相联(Fully-Associative)允许 NRegion 中的一个块映射到 PCache 的任意位置,如图 6(b)所示,存储区域含有 n 个带,每个带的数据块都映射到缓存 C_0 中的任意位置。它的优点是灵活,缓存利用率高,缺点是缓存查找速度慢。全相联实际上可以看成一种特殊的组相联,其中 $m = 1$。在 SSWD 中 PCache 与 NRegion 之间的映射方式就是全相联[9]。

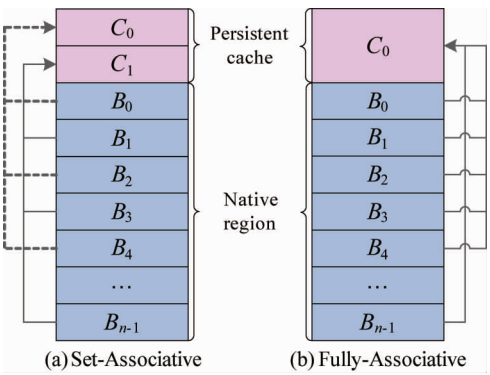

图 6　全相联和组相联

采用组相联映射,当组内有效缓存块数目达到阈值后就会触发缓存回收;而对于全相联,只有当整个缓存的剩余空闲空间达到阈值时才触发缓存回收。较全相联,组相联造成的缓存回收的次数会更多。在 HSWD 中缓存冲突造成的影响会被放大,因为其回收方式需要执行非常耗时的 RMW 操作。同时,若多个组都需要回收空间,而在同一时间只能回收一个组,则其它组必须等待,从而对那些等在其它组上的 IO 请求造成非常大的延迟;而全相联不同,一旦有空间被回收,IO 请求就能立即被响应。

相对于全相联缓存查找速度慢对系统性能造成的影响,组相联频繁的 RMW 操作造成的影响更大。本文分别实现并对比了上述两种映射方式,对比结果也验证了上述观点。因此 HSWD 最终选择了全相联方式,这亦可解释为何 SSWD 选择全相联方式。

2.3　数据布局

与 SSWD 类似,NRegion 按 Band 组织划分。而与 SSWD 中 PCache 的数据组织方式不同,HSWD 中 PCache 划分为三个区域:元数据区,缓存数据区和预留缓存区。

元数据区:包括 HSWD 的元数据和 PCache 中缓存块的元数据。前者包括 PCache 大小、缓存块大小、所采取的回收策略等;后者包括缓存块对应的磁盘块号及状态信息。存储元数据的目的是为了保证数据的正确性以及断电之后的数据恢复。

缓存数据区:由一系列的缓存块组成。对于组相联,该区域被划分为多个组,每个组由多个缓存块组成;而对于全相联,相当于只有一个组。

预留缓存区:执行 RMW 的临时区域。缓存回收执行 RMW 时,首先将脏块所在带的全部数据读到预留缓存区,然后使用缓存中的脏数据来修改预留缓存区的数据,最后将修改后的整带数据写回 NRegion。在预留缓存区而不是直接在内存中执行 RMW 操作的主要原因是:如果在内存中执行 RMW 过程中断电,一方面会导致数据丢失,另一方面会使得带内数据不一致,而在预留区域中执行 RMW 则不会产生上述问题。

3　HSWD 典型流程

3.1　读流程算法

算法 1 描述了 HSWD 读流程。

算法 1: HSWD Read

```
Function HSWD _ Read (LBA)
1:  band = get _ band (LBA)
2:  set = band% M
3:  block = lookup _ in _ PCache (LBA, set)
4:  if  block! = NOT _ IN _ CACHE  then
5:      read _ from _ PCache (block, set)
6:  else
7:      if is _ RMWING (band)  then
8:          read _ from _ RCache (LBA)
9:      else
10:         read _ from _ NRegion (LBA)
11:     end if
12:  end if
```

读一个块时,首先根据逻辑块地址确定其在 PCache 中所在的组。若读请求命中 PCache,则直接从 PCache 读取。若不命中,通常直接从 NRegion 中读取,但当要读取的块所在的带正在执行 RMW 的

W 操作时,则必须等到 RMW 完成之后才能读取,这就增大了读延迟。但此时在预留缓存区中数据一直是正确的,所以可直接从预留缓存区中读取,这就使得 RMW 操作和读操作并发,提高了性能。

3.2 写流程算法

算法 2 描述了 HSWD 的写流程。写一个块时,首先判断块是否命中 PCache,若命中则直接写入 PCache;否则,找一个空闲的缓存块,如果没有空闲的缓存块,将触发 PCache 的回收以获得可用的缓存块,然后写到这个缓存块中。最后检查 PCache 中已使用的缓存块数目是否超过了设定的触发回收的阈值,若是,将触发 PCache 的回收。

算法 2: HSWD Write

```
Function HSWD _ Write (LBA)
1:  SET _ CLEAN _ THRESHOLD = 80%
2:  band = get _ band (LBA)
3:  set = band % M
4:  block = lookup _ in _ PCache (LBA, set)
5:    if block! = NOT _ IN _ CACHE then
6:      write _ to _ PCache (block, set)
7:  else
8:      block = get _ free _ block (set)
9:      while block = = NO _ ROOM do
10:       HSWD _ clean _ set (set)
11:       block = get _ free _ block (set)
12:     end
13:     write _ to _ PCache (block, set)
14:  end if
15:  if Dirty _ block _ percent > = SET _ CLEAN _
     THRESHOLD then
16:     HSWD _ clean _ set (set)
17:  end if
```

3.3 PCache 回收算法

PCache 通过执行 RMW 操作下刷脏数据来回收缓存空间。回收流程如算法 3 所示,首先会选择一个 Band,然后对这个 Band 执行 RMW 操作,最后检查已经使用的块的数目是否小于设定的停止回收的阈值,若是则停止回收。

算法 3: HSWD Clean Pcache Set

```
Function HSWD _ clean _ set (set)
1:  STOP _ CLEAN _ THRESHOLD = 60%
2:  band = select _ band (set)
3:  RMW (band)
4:  if Dirty _ block _ percent > = STOP _ CLEAN _
    THRESHOLD then
5:      HSWD _ clean _ set (set)
6:  end if
```

RMW 流程如算法 4 所示。Read _ to _ RCache 将待回收 Band 的全部数据读到预留缓存区中;Modify _ with _ PCache _ blocks 使用属于这个 Band 的 PCache 中 blocks 来修改预留缓存区中 Band 的数据,修改完成后,就可以将这些 PCache 中的 blocks 设置为无效,这样就回收了空间;最后将预留缓存区中的全部数据写回 NRegion。

算法 4: RMW

```
Function RMW (band)
1:  Read _ to _ RCache (band)
2:  Modify _ with _ PCache _ blocks (band)
3:  Invalidate _ PCache _ blocks (band)
4:  Write _ to _ NRegion (band)
```

3.4 Band 选取策略

在 HSWD _ clean _ set 算法中 select _ band 选取用于执行 RMW 操作的带。在 SSWD 中,受限于 PCache 的管理方式与 SMR 的叠瓦特性,使得 FIFO(先进先出)策略是必然选择。但在 HSWD 中,PCache 灵活的数据组织方式使得 Band 选取策略更加灵活。在 HSWD 中设计并实现了三种:FIFO,LRU(最近最少被访问),MOST(含有缓存块最多)。

FIFO 策略仅按照 Band 被写的先后顺序进行选取,不考虑数据的热度、Band 中脏块的数目等与回收效率相关的因素。MOST 策略是比较贪婪的策略,其选取含有缓存块最多的 Band,使得执行一次 RMW 回收的缓存块的数目最多,特别是当缓存压力比较大时,会有较好的效果,但其不考虑数据的热度,这可能导致某个 Band 被频繁的执行 RMW,从而

影响回收效率;若此时采用 LRU 策略,能有效减少执行 RMW 的次数。但其并不考虑执行一次 RMW 可以回收的缓存块数目,这使其回收效率较低,当缓存压力较大时,回收相同的缓存块,需要比 MOST 策略执行更多的 RMW。虽然 MOST 策略一次回收的缓存块最多,但其耗时也最长。

从缓存为空,到被填满,再到被清空的整个过程来看,LRU 策略执行 RMW 次数最少,这主要归因于其避免对较热的 Band 重复执行 RMW,这是一种考虑未来效率多过当前效率的策略;而 MOST 策略尽可能地将执行一次 RMW 回收较少空间的 Band 推后去回收,这是一种只考虑当前效率的较为激进的策略。但大多数情况下 MOST 策略比 LRU 策略更有效,因为 MOST 策略在提升当前回收效率的同时推后了 RMW 效率不高的 Band 的回收,而这些 Band 的回收很有可能在空闲时间内完成,并不影响系统的性能。在后续测试对比在不同 Band 选取策略下系统的性能,并验证上述观点。

4 测试结果与分析

4.1 测试环境

HSWD 使用 Device Mapper 机制实现为一个 Linux 内核模块,并提供标准的块设备接口。在 HSWD 中,PCache 使用 120GB SSD 的部分空间来模拟,具体使用大小在测试中指定;而 NRegion 使用 2TB HDD 进行模拟,并且为了模拟的准确性,关掉了硬盘的写缓存。SSWD 的带大小介于 15MB ~ 40MB 之间,由内而外逐渐增大[9],而 HSWD 的带大小为固定值,为保证对比的准确性,设置为与 SSWD 带大小相近的 32MB,且设置预留缓存区大小为带大小。具体的实验平台参数设置如见表 1。

4.2 测试工作负载

测试工作负载主要包括两类:基准测试程序 FIO 生成的工作负载和实际应用中典型工作负载。测试目的是针对 HSWD 的随机写性能进行评测,因此 FIO 生成的工作负载为随机写,并选取写请求比例高的典型工作负载[19],其特征在表 2 中列出。

表 1 实验平台设置

项目	具体配置	
CPU	Pentium(R) Dual-Core CPU E6600 @ 3.06GHz	
OS	CentOS release 6.5; kernel version 2.6.32.68, x86_64	
HSWD	Size	2TB
	PCache	Intel SSD DC S3500 Series 120GB
	NRegion	Seagate SATA 128MB Cache 6Gb/s 7200rpm 2TB HDD (write cache disable)
	PCache block size	4kB
	Band size	32MB
SSWD	Seagate SATA 6Gb/s NCQ 8TB SWD	

在测试 HSWD 中 PCache 不同回收策略以及不同容量对系统性能的影响时,为方便测试和更明显地看出效果,选取写数据量和 Footprint 相对较小的工作负载: mds0,prxy0,rsrch0,src2_0。在与 SSWD 和 Flashcache 进行性能对比的测试中,选取写数据量大且 Footprint 较大的工作负载:prn0,hm0,proj0。

4.3 组相联与全相联对比测试

使用 FIO 测试对比全相联和组相联,其配置为 iodepth = 31,使用 libaio 方式随机写 2GB 的数据,写请求块大小为 4kB。本测试中,HSWD PCache 容量设置为 256MB,当带的大小为设置为 32MB 时,预留空间为 32MB,实际的数据缓存空间为 224MB,对于组相联,共有 7 个组。

表 3 列出了在全相联和组相联对比的测试结果,可以看出:全相联的性能高于组相联。较其它参数,两者的最大 IO 延迟相差非常大,全相联仅为 4s,而组相联高达 26s。观察测试过程中每秒进行读写(I/O)操作的次数(input/output operations per second, IOPS)的变化,刚开始两者的 IOPS 均维持在很高的水位,随着 PCache 被填满,后续 IO 请求需等待缓存执行 RMW 操作回收空间,导致 IOPS 下降。

表 2 典型工作负载特征

Work-load	Total Request	Write Request	Write percent(%)	Write Data(GB)	Foot-print (GB)	Avgerage Request Len(B)
mds0	1200000	1057173	88.1	7.3	0.33	9440
prxy0	1200000	1150891	95.9	2.6	0.21	2574
rsrch0	1200000	1089630	90.8	9.1	0.27	9182
src2 _0	1200000	1051556	87.6	7.1	0.46	7395
prn0	5585886	4983406	64.5	46.0	12.4	11358
hm0	3993316	2575568	89.2	20.5	1.6	8185
proj0	4224524	3697143	87.5	144.0	1.7	38948

表 3 组相联和全相联测试结果

Policy	Avgerage IOPS	Avgerage IO Latency	Max IO Latency	RMW times	Runtime
Fully	483	64.1ms	4.0s	274	18.1min
Set	387	80ms	26.2s	351	22.6min

图 7 展示以 4s 为间隔 IOPS 的变化情况，去掉最开始几秒 IOPS 过高的情况，可以看出，全相联(Fully-Associative)的 IOPS 相对比较稳定，维持在 400 左右；而组相联(Set-Associative)，其 IOPS 抖动非常大，一段时间内 IOPS 接近 0，然后 IOPS 又升高，如此反复。这与前文理论分析的结果吻合。在 HSWD 的实现中，持久缓存的缓存块的状态信息在内存中维护，并采取高效的哈希方式进行查找和访问，对于全相联来说，持久缓存在寻址和访问过程产生的性能开销远小于执行 RMW 操作回收空间所产生的性能开销，所以该部分开销可忽略不计。因此，HSWD 最终选择全相联方式。

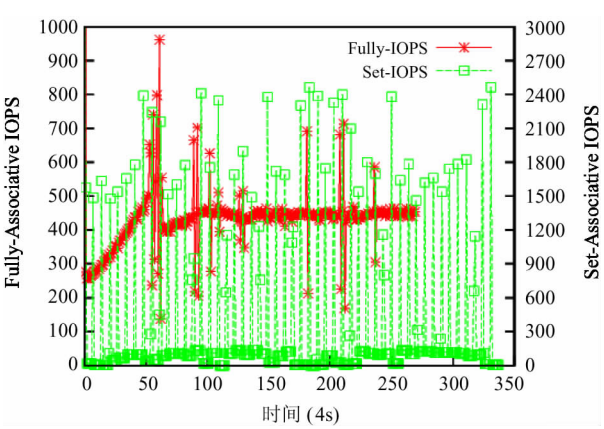

图 7 不同映射策略下 4s 间隔内 IOPS 的变化规律

4.4 不同 PCache 回收策略对比测试

当 PCache 中脏数据量超过设定的阈值时，需要启动回收操作。本节通过回放四个工作负载：mds0，prxy0，rsrch0，src2 _0 来对比在不同回收策略下 HSWD 的性能。测试过程中，PCache 容量设置为 256MB，并使用全相联的映射策略。

图 8 统计平均 IO 响应时间，可以看出 MOST 策略的平均 IO 响应时间要低于另两种策略。图 9 统计在不同策略下执行 RMW 的次数，对于 mds0，prxy0 和 rsrch0，LRU 和 FIFO 策略的平均 IO 响应时间相差不大，但对 src2 _0，LRU 策略的平均 IO 响应时间非常大，主要原因为 LRU 策略产生的 RMW 次数远高于另两种策略。这说明 RMW 次数是决定 HSWD 性能的重要因素。而 RMW 的次数主要受两个因素的影响：执行一次 RMW 回收的缓存块数和缓存写请求的命中率。图 10 统计缓存写请求的命中率，可以看出采用 LRU 策略的缓存写请求命中率

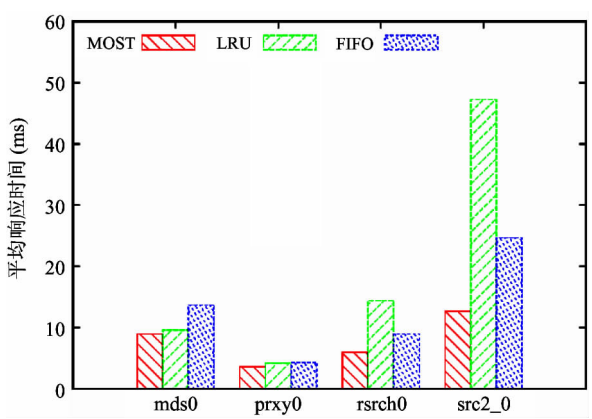

图 8 不同缓存策略下平均 IO 响应时间

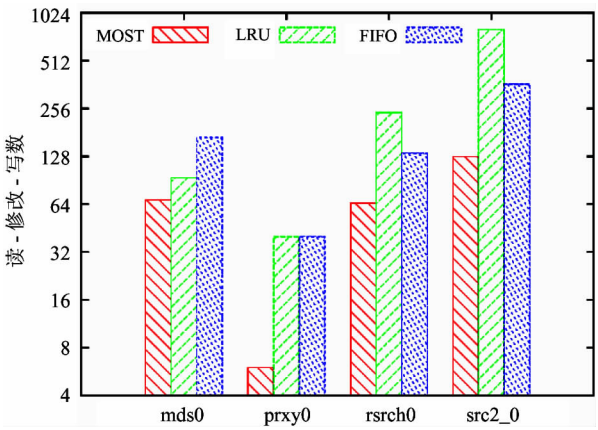

图 9　不同缓存策略下执行 RMW 的次数

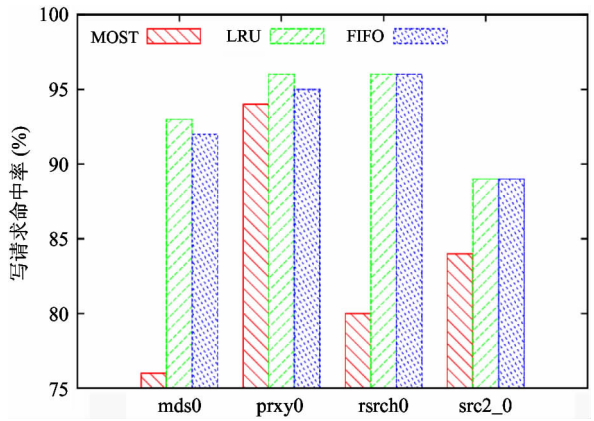

图 10　不同缓存策略下缓存写请求的命中率

最高,而采用 MOST 策略的缓存写请求命中率最低,但采用 LRU 策略的性能反而比 MOST 策略低,说明在这个测试中,第一个因素要比第二个因素对 RMW 次数的影响更大。

图 11 以 10000 个请求为单位,统计回放 src2 _ 0 过程中平均响应时间的变化,虽然采用 MOST 策略的平均响应时间最低,但从图 11 中可看到:在某些请求段内 MOST 策略的平均响应时间反而比另两种策略高。可能的原因有两个:其一,对局部性较好的数据访问,采用 MOST 策略使得某个带被频繁的执行 RMW,而实际的回收效率并不高;其二,虽然 MOST 策略一次回收的缓存块最多,但其耗时也最长,而恰好此时执行一次 RMW 的时间对性能的影响较大。

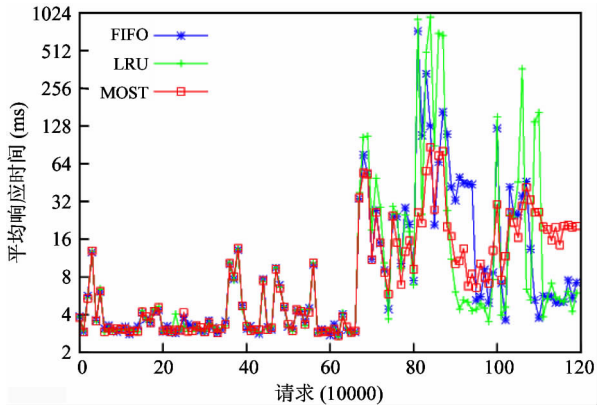

图 11　不同缓存策略下回放 src2 _ 0 过程中平均响应时间变化

4.5　PCache 容量对 HSWD 性能的影响

本节通过回放 mds0,prxy0,rsrch0,src2 _ 0 测试当 PCache 容量设置为 128MB/256MB/384MB 时 HSWD 的性能。图 12 统计测试过程中平均 IO 响应时间;图 13 统计了测试过程中执行 RMW 的次数。

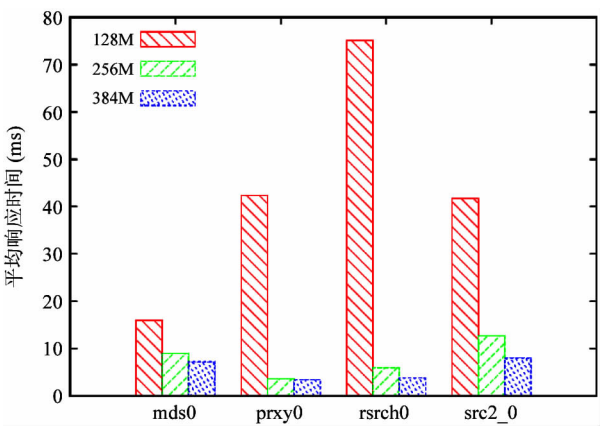

图 12　不同 PCache 容量下平均 IO 响应时间

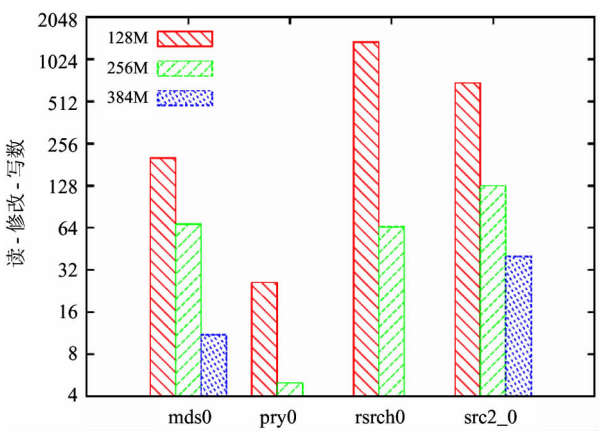

图 13　不同 PCache 容量大小下执行 RMW 的次数

从图 12 和图 13 可以得出如下结论:PCache 容量越大,执行 RMW 次数越少,HSWD 的性能越高。而难点在于如何根据不同的应用需求设定合理的 PCache 容量。从图 12 可以看到 PCache 容量从 128MB 增大到 256MB 时性能提升明显,而从 256MB 增大到 384MB 时,性能提升幅度较小。从图 13 可找到原因,当 PCache 容量从 128MB 增至 256MB 时,RMW 次数明显下降,而容量从 256MB 增至 384MB 时,下降相对较小,当缓存达到 384MB 时,对于 prxy0 和 rsrch0 已经没有 RMW 操作。

4.6 HSWD 与 SSWD/Flashcache 性能对比测试

本节测试中,HSWD 使用全相联映射和 MOST 缓存回收策略,并通过回放 prn0, hm0, proj0 对性能进行评测。设置 HSWD 的 PCache 容量为 1GB。原因有两个:其一,从表 2 可以看出这三个工作负载的写数据量分别为 46GB、20.5GB、144GB,且它们的 Footprint 也分别达到了 12.4GB、1.6GB、1.7GB;其二,SSWD 中 PCache 大小约为 100GB[10],1GB 的 SSD 容量相对于 100GB 的瓦记录磁盘容量从价格以及对比的公平性上来说,也是合理的。同时,Flashcache 中设定缓存大小也为 1GB。

图 14 为该测试的结果,从图中可以很明显的看出,HSWD 的性能表现优于 SSWD 和 Flashcache。以 hm0 为例,HSWD 的性能较 SSWD 提升 2.25 倍,较 Flashcache 提升 1.32 倍。尽管 HSWD 和 SSWD 均采用全相联的映射策略,但是对于 SSWD,将需要更长的时间进行缓存回收,这主要归因于其采用的 FIFO策略;而对于Flashcache,因为其采取组相联的映射策略,使得其需要执行回收的次数远多于 HSWD。因此,HSWD 能够提供比 SSWD 和 Flashcache 更好的性能。

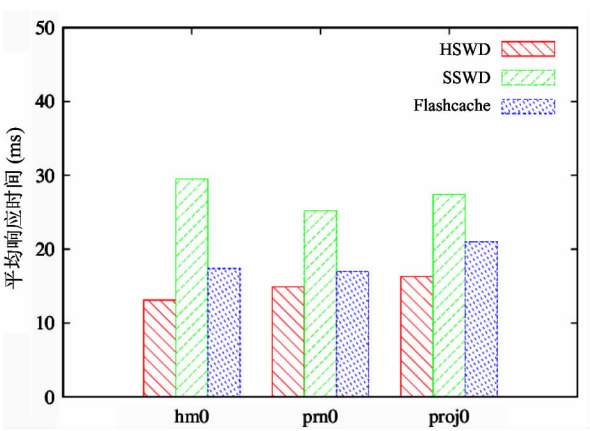

图 14 HSWD/SSWD/Flashcache 的平均响应时间

5 结 论

为解决瓦记录磁盘持续随机写入性能较低的问题,本研究设计并实现了使用固态硬盘(SSD)作为持久缓存的混合瓦记录磁盘(HSWD),设计并实现了两种不同的持久缓存到本地存储的映射方案——全相联和组相联,以及三种不同的持久缓存回收策略:最近最少被访问(LRU)、先进先出(FIFO)和含有缓存块最多(MOST),并通过实验和理论分析分别对比了两种映射方案、三种缓存回收策略,以及持久缓存的大小对系统性能的影响。实验结果表明:全相联因能最大化使用持久缓存空间并减少回收次数而优于组相联;在多数情况下,较 LRU 和 FIFO 策略,MOST 策略因一次回收的持久缓存空间最多而更适合于 HSWD;轻微的持久缓存容量提升就使得 HSWD 的性能大幅提升,关键在于依据不同工作负载的特点设定合理的持久缓存容量大小。上述实验结果与理论分析结果基本吻合。最后,通过回放典型工作负载对比 HSWD,SSWD 和 Flashcache,测试结果表明:HSWD 的性能明显优于 SSWD 和 Flashcache。

目前,HSWD 中 Band 选取策略在系统初始时设定且运行过程中不能更改,期望后续可以根据数据访问特征实时动态调整。此外,SSD 的稳定性和寿命问题也没有在本文中考虑到,这也是后面设计中要考虑的一个问题。

参考文献

[1] Wood R, Williams M, Kavcic A, et al. The feasibility of magnetic recording at 10 terabits per square inch on conventional media. *IEEE Transactions on Magnetics*, 2009, 45(2): 917-923

[2] Richter H J, Dobin A Y, Heinonen O, et al. Recording on Bit-Patterned Media at Densities of 1Tb/in and Beyond. *IEEE Transactions on Magnetics*, 2006, 42(10): 2255-2260

[3] Kryder M H, Gage E C, Mcdaniel T W, et al. Heat assisted magnetic recording. *Proceedings of the IEEE*,

2008, 96(11): 1810-1835
[4] Zhu J G, Zhu X. Microwave assisted magnetic recording. *IEEE Transactions on Magnetics*, 2008, 44(1): 125-131
[5] Greaves S, Kanai Y, Muraoka H. Shingled recording for 2-3 Tbit/in^2. *IEEE Transactions on Magnetics*, 2009, 45(10):3823-3829
[6] Seagate Archive HDD. http://www.seagate.com/products/enterprise-servers-storage/nearline-storage/archive-hdd: Seagate, 2015
[7] Feldman T, Gibson G. Shingled magnetic recording: areal density increase requires new data management. *The magazine of USENIX&SAGE*, 2013, 38(3): 22-30
[8] Gibson G, Ganger G. Principles of operation for shingled disk devices. *Electric Machines & Power Systems*, 2011, 5(6):485-496
[9] David H, John H M, Jonathan D C. Data handling algorithms for autonomous shingled magnetic recording HDDs. *IEEE Transactions on Magnetics*, 2012, 48(5): 1777-1781
[10] Aghayev A, Shafaei M, Desnoyers P. Skylight—a window on shingled disk operation. *Acm Transactions on Storage*, 2015, 11(4):1-28
[11] Maraldo P. Directions for shingled-write and two-dimensional magnetic recording system architectures: synergies with solid-state disks. *Parallel Data Lab*, 2009, 6(3): 514-520
[12] Cassuto Y, Sanvido M A A, Guyot C, et al. Indirection systems for shingled-recording disk drives. In: Proceedings of the MASS Storage Systems and Technologies, Incline Village, USA, 2010. 1-14
[13] Amer A, Long D D E, Miller E L, et al. Design issues for a shingled write disk system. In: Proceedings of the IEEE, Symposium on MASS Storage Systems and Technologies, Incline Village, USA, 2010. 1-12
[14] Lin C, Park D, He W, et al. H-SWD: incorporating hot data identification into shingled write disks. In: Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Washington, USA, 2012. 321-330
[15] Luo D, Wan J, Zhu Y, et al. Design and implementation of a hybrid shingled write disk system. *IEEE Transactions on Parallel & Distributed Systems*, 2015, 27(4):1-1
[16] Suresh A, Gibson G, Ganger G. Shingled magnetic recording for big data applications. *IEEE Transactions on Magnetics*, 2012, 47(10): 3691-3697
[17] Moal D L, Bandic Z, Guyot C. Shingled file system host-side management of shingled magnetic recording disks. In: Proceedings of the IEEE International Conference on Consumer Electronics, Las Vegas, USA, 2012. 425-426
[18] Jin C, Xi W Y, Ching Z Y, et al. HiSMRfs: a high performance file system for shingled storage array. In: Proceedings of the MASS Storage Systems and Technologies, Santa Clara, USA, 2014. 1-6
[19] SNIA IOTTA Repository. MSR Cambridge Block IO Traces. http://iotta.snia.org/traces/list/BlockIO: Snia, 2015

Design and implementation of a high performance hybrid shingled write disk system

Ma Liuying[*,**], Xiao Wenjian[*,**], Dong Huanqing[*], Xu Lu[*]
([*]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
([**]University of Chinese Academy of Sciences, Beijing 100049)

Abstract

The shingled magnetic recording (SMR), a new magnetic recording technique, was analyzed, and the following were pointed out: the main drawback of the SMR is that random-write access to the disk is restricted due to the overlaps in the layout of data tracks; The seagate shingled write disk (SSWD) mitigates the random-write access restrictions by using an internal persistent cache, but it encounters the poor performance with sustained random writes. In consideration of the situation, a hybrid shingled write disk (HSWD) system was designed by combining it with the solid state drive (SSD). Then, two different policies for caching between the persistent cache and the native region, called the fully-associative and the set-associative), as well as the three different collection policies of LRU, FIFO and MOST, were implemented, and they were compared respectively by some experiments and theoretical analysis. The performance of the HSWD was tested and compared with the SSWD and Flashcache. The evaluation results demonstrated that the performance in persistent random writing of the HSWD was great higher then the SSWD and Flashcache, for example, 2.25x faster than the SSWD, and 1.32x faster than Flashcache in replaying hm0 trace.

Key words: shingled magnetic recording (SMR), seagate shingled write disk (SSWD), solid state drive (SSD), hybrid system, caching policy, collection policy