



一种 SSD 存储系统中的先应式垃圾回收算法

屠雪真¹, 黄震江², 陈正光²

(1. 河南大学计算机与信息工程学院, 河南 开封 475001;

2. 中兴通讯股份有限公司, 江苏 南京 210012)

摘要: 提出了一种基于预测的先应式垃圾回收算法, 首先根据不同的热度因子对数据进行读、写热度分离, 然后对未来将达到的不同类型的页面分配请求 (page allocation request, PAR) 数量进行上位和下位预测, 在满足页面分配请求 PAR 下位预测的同时, 最大程度地满足 PAR 上位预测需求, 优化了 WA 问题, 减少了不必要的有效数据迁移, 从而最大化垃圾回收效用。对此问题定义了数学模型, 并给出了一种获取近似最优解的算法, 对模型适用场景进行了分析。实践结果表明, 本算法能够获取最大收益, 提升 SSD 读写性能并降低成本。

关键词: SSD; 垃圾回收; 数据迁移; 写入放大; 碎片整理

中图分类号: TP333

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2019075

Provident garbage collection algorithm for SSD storage system

TU Xuezheng¹, Huang Zhenjiang², CHEN Zhengguang²

1. School of Computer and Information Engineering, Henan University, Kaifeng 475001, China

2. Zhongxing Telecommunication Equipment Corporation, Nanjing 210012, China

Abstract: A predictive based proactive garbage collection algorithm was proposed. Firstly, the data was separated according to different heat factors, then the upper and lower predictions were performed on the number of different types of page allocation requests (PAR) that would be reached in the future. While satisfying the page allocation request lower prediction, the PAR upper prediction requirement was maximally satisfied, the WA problem was optimized, and invalid effective data migration was reduced, thereby maximizing the garbage collection utility. A mathematical model was defined for this problem, and an algorithm for obtaining the approximate optimal solution was given. The applicable scenario of the model was analyzed. The practical results show that the algorithm can obtain the maximum benefit and can significantly improve the performance of SSD and reduce the cost.

Key words: solid state drive, garbage collection, data migration, write amplification, defragmentation

收稿日期: 2018-12-29; 修回日期: 2019-04-12

基金项目: 科技部云计算和大数据重点专项基金资助项目 (No.2018YFB1003300); 深圳市科技计划项目 (No.SF20170036)

Foundation Items: Ministry of Science and Technology Cloud Computing and Big Data Key Project (No.2018YFB1003300), Shenzhen Science and Technology Plan Project (No.SF20170036)

1 引言

与传统机械硬盘比较,固态硬盘(solid state drive, SSD)有着读写速度快、防震抗摔性好、低功耗、无噪音、重量轻、体积小等优势^[1]。因此 SSD 在许多领域已经开始替代传统硬盘,它是当前存储领域的研究热点之一。

目前的 SSD 主要是采用闪存(flash 芯片)作为存储介质,而闪存的结构与传统的磁存储介质不同,其主要特点有:闪存只提供读、写和擦除 3 种操作,且这 3 种操作性能不对称,读最快,写次之,擦除最慢。闪存是按页(page)、块(block)、平面(plane)的结构进行组织的,页是读/写的最小单位,一般为 2 KB、4 KB、8 KB;块是擦除的最小单位,一个块一般包含 64 页、128 页;1 个闪存单元在经过 1 次写入操作后,必须先对该闪存单元进行 1 次擦除操作,才能够写入新数据,即所谓的写前擦除(erase-before-write)^[2],这造成闪存无法像传统硬盘一样直接覆盖写。

随着 SSD 的持续使用,一个数据块中往往含有 3 种类型的页面:可写页面(writable page, WP)、有效页面(valid page, VP)、垃圾页面(garbage page, GP)。其中可写页面是该块中可以直接写入数据的页面,有效页面是指被系统使用且已经写入有效数据的页面,垃圾页面是指该页面写入的数据不再被系统使用,需要经过块擦除后才能再次供系统使用的页面。如图 1 所示,一个数据块包含 32 个页面。当要擦除的数据块 B2 包含有效页面 VP4 时,需要先将有效页面 VP4 的数据搬迁到一个或者几个块,例如 B1,然后再把不再包含有效页面的数据块 B2 进行擦除来释放空间。因此,垃圾回收(garbage collection, GC)成为 SSD 区别于传统磁盘的重要特征。

此外,闪存每个存储单元的编程/擦除(P/E)次数有限,超过 P/E 次数后,该单元存储数据不再可靠,因此 SSD 盘的擦除次数是有限的。显然,

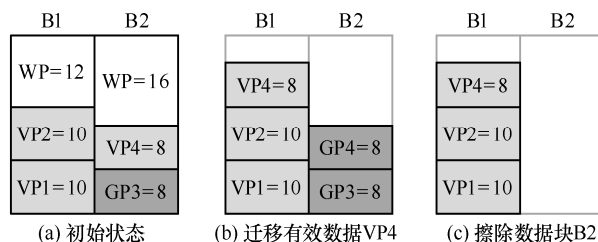


图 1 SSD 垃圾页面回收示意图

提高回收垃圾页面的效用,减少不必要的数据迁移数量,可以增加 SSD 磁盘的使用寿命。

2 相关工作

由于闪存的这些独特的性质和操作,为了屏蔽闪存和传统磁盘不一样的读写特性,以适应当前的文件系统,需要一个专用软件层来对闪存进行管理和优化,这就是闪存转换层(flash translation layer, FTL)^[3]。它一般包括地址映射、磨损平衡和垃圾回收 3 个模块,闪存转换层的优劣极大影响到闪存的性能和寿命。

DFTL(demand-based FTL)^[4]是经典的闪存转换层算法。它把较大的全局映射信息表存放在闪存中,只将常用的映射项放在缓存中,根据映射项访问的近期性作为缓存替换的依据,解决了页级映射策略中映射信息较大和缓存容量有限的矛盾。但是,DFTL 没有充分利用负载的空间局部性特点,在缓存失效时频繁的脏映射项换出会导致大量的映射页写操作;它也未能优化垃圾回收过程中有效页迁移导致的写入放大问题^[5]。

闪存的垃圾回收过程是造成写放大问题的主要因素。在垃圾回收过程中,若回收块内有未被覆写的有效数据页,则必须先将有有效页迁移走,需要至少 1 次读操作和 1 次写操作来完成迁移过程。此外,有效页的迁移还会导致其对应的逻辑地址和物理地址映射信息发生改变,从而引起额外的元数据读写操作。被回收物理块内的有效页越多,则需要迁移的有效页也越多,对闪存性能和寿命带来的负面影响也更大。



在 DFTL 思想的启发下, 国内外的研究人员提出各种新型页级闪存转换层算法。例如, 姚英彪等人^[6]提出一种基于分类策略的聚簇页级闪存转换层算法 CPFTL, 将地址映射缓存分为热、冷、连续 3 类来提升各类请求的处理能力。算法 S-FTL^[7]提出通过识别 3 种空间本地性访问模式来减少页映射大小。张琦等人^[8]也提出了一种对 DFTL 的优化方法, 它能够给设计基于转换页的数据聚集技术, 将地址映射信息属于同一转换页的数据写入相同的数据块, 降低了垃圾回收时转换页的更新数目, 提高了系统性能。

在垃圾回收方案上, 现有方案主要分为被动式回收和主动式回收两类。被动式回收策略在页面分配请求 (page allocation request, PAR) 到达后, 判断能否实现分配。如果不能, 则将对一个或多个数据块进行有效数据迁移和擦除数据块的操作, 以回收垃圾页面腾出空间来满足新到达的 PAR。当短时间内大量 PAR 无法满足时, 被动式回收策略将发生频繁的有效数据迁移、擦除数据块操作, 导致 SSD 性能下降和服务延迟。

现有主动式回收策略, 主要是通过整合垃圾页面, 最小化包含垃圾页面的数据块数量, 腾出空间以满足后续页面分配需求。如图 1 (c) 所示, 通过迁移 VP4, 擦除数据块 B2, 可以满足新到达的大小为不超过 32 个页面的 PAR。此技术可在一定程度上减少垃圾页面碎片, 但容易产生频繁且不必要的有效数据迁移。图 1 (a) 中, 当新到达两个 PAR 大小分别为 12 个和 16 个页面时, 则无需整合迁移, 可以直接实现分配。

此外, 还有一些研究通过分析垃圾页面产生的原因, 期望通过热度预测的方法, 避免产生垃圾页面碎片, 从而避免垃圾回收时的有效数据搬迁。在热度评估及应用方面, 现有大部分算法实质是对 LRU、LFU 等经典算法的局部改进与重新组合。

LRFU (least recently and frequently used) 算法^[9], 通过结合最不经常使用 (least frequently used, LFU) 算法和最少使用 (least recently used, LRU) 算法实现。LRFU 可以作为 LFU 使用, 也可以作为 LRU 使用, 在时效场景和频率场景下应用 LRFU 算法都能取得较好的效果。

ARC (adjustable replacement cache) 算法^[10], 根据数据块被访问的次数分离存储, 将存储层分为一次访问存储和多次访问存储两部分, 一次访问存储只存储仅访问过一次的数据块。为提升命中率, ARC 根据具体的数据访问 workflow 自动调整这两部分存储空间的大小。

LIRS (low inter-reference recency set, LIRS) 算法^[11], 计算两次访问同一数据块之间的距离, 据此动态排序访问数据块。LIRS 算法将每个存储层划分为 LIR 空间和 HIR 空间两部分, 并保证 LIR 空间里的所有数据块都比 HIR 空间里的数据块具有更小的访问间隔。

基于缓存映射项重用距离的地址映射方法^[12], 实现写缓存映射表的冷热分区并采取不同的管理策略, 减少映射页写操作, 同时实现了基于重用距离的冷热数据分离存储, 提高垃圾回收效率。

综上所述, 一个有效的 SSD 垃圾回收算法应该关注: 何时执行必要的垃圾回收操作? 在垃圾回收过程中, 哪些数据块将被擦除回收? 针对被擦除回收数据块中的有效数据, 如何迁移? 此问题是 NP 难问题。

为此, 本文提出了一种基于预测的先应式垃圾回收算法 (a proactive garbage collection algorithm based on prediction, PGCABP)。先根据一种非对称读写热度算法划分 SSD 存储区域, 在给定存储区域中现有 PAR 集合的条件下, 对后续将到达的 PAR 集合进行预测, 据此求出最大程度避免无效的有效数据迁移的垃圾回收算法, 即获取近似最优解的方法。

3 PGCABP 算法设计

3.1 问题定义

基于 SSD 块页二级存储物理特性分析可以知道,造成 SSD 垃圾页面的主要因素是存储数据发生了变化,因此,分离容易发生改变的数据,可以提升 SSD 的整体性能。基于此,将 SSD 划分为初始区、热读区、热写区 3 个存储区域。如图 2 所示,数据首先保存到初始区,当初始区/热读区数据发生改变时,则将数据移入热写区;初始区在发生数据迁移时,对被迁移的数据进行热度评估,被评估为热度数据时,将数据迁移到热读区,否则在初始区内部迁移;热读区/热写区在发生数据迁移时,对被迁移的数据进行热度评估,当不再被评估为热度数据时,将数据迁移到初始区,否则在热读区/热写区内部迁移。对于热度评估,采用非对称读写热度(asymmetric effects of read and write heat, AERWH)算法,对 PAR 数据块 m 在时刻 t 的热度 $H(m,t)$ 进行评估:

$$H(m, t) = v_r \cdot h_r(m, t) + v_w \cdot h_w(m, t) \quad (1)$$

其中, v_r 为读热度因子, v_w 为写热度因子,对参数 v_r 、 v_w 进行非对称假定,即两者取值有显著差异。 $h_r(m, t)$ 、 $h_w(m, t)$ 可以通过 LRU、LFU、ARC 等算法评估。

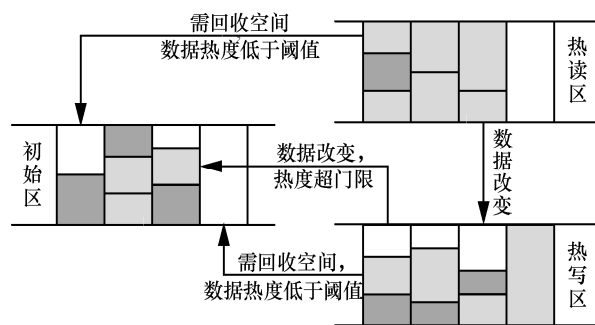


图2 分区存储示意图

这样,通过 AERWH 算法评估读写热度对数据进行分区存储,可以集中垃圾页面在数据块中的分布,增加垃圾回收的效率。同时,对初始区、

热读区、热写区的垃圾回收管理,都可以看作在存储区域既定 PAR 分配条件下,如何满足新到达的 PAR 需求,可以归一成同一模型讨论。

给定一个具有 B 个可用数据块的存储区域,其中,对于数据块 b ($1 \leq b \leq B$): S 为该数据块所含数据页总量, V_b 为有效页面数量, G_b 为垃圾页面数量, W_b 为可写页面数量,则有 $S = V_b + G_b + W_b$ 。假设存储区域受理 C 种类型的 PAR, c -PAR 表示类型为 c 的 PAR。每个 c -PAR 可以用一个元组 $\langle c, u_c, s_c \rangle$ 表示,其中 c 为 PAR 类型索引, u_c 为 c -PAR 效用收入因子, s_c 为 c -PAR 请求的页面数量。一般地,假定 $s_1 > s_2 > \dots > s_C$, PAR 请求的页面数量越多,则效用收入因子越高,所以有 $u_1 > u_2 > \dots > u_C$ 。

在给定的时间点 t ,存储区域可存在已经分配资源的 PAR(包括有效的和垃圾的),在该条件下,讨论满足时刻 t 新到达 PAR 的垃圾回收最佳分配。另外,可以对未来一段时间 $[t, t']$ 新到达的 PAR 数量进行预测,从而在时间点 t 求解为满足未来一段时间 $[t, t']$ 新到达的 PAR 的分配需求、SSD 垃圾回收的迁移决策,以最大化 SSD 垃圾回收效用。

3.2 满足时刻 t 新到达 PAR 的垃圾回收最佳分配

考虑满足时刻 t 新到达 PAR 的垃圾回收最佳分配问题,在时间点 t ,采用 $m_c \times B$ 0-1 矩阵 $A'(c)$ 记录存储区域现有 c -PAR 的分配情况:矩阵元素 $a(c)_{rb} = 1$ 表示第 r 个 c -PAR 在数据块 b 分配空间,否则 $a(c)_{rb} = 0$ 。采用 m_c 维 0-1 向量 $D'(c)$ 记录存储区域现有分配的 c -PAR 在 t 时刻是否已转变为垃圾页面情况: $d(c)_r = 1$ 表示第 r 个 c -PAR 分配页面转为垃圾页面,否则 $d(c)_r = 0$ 。采用 $n_c \times B$ 0-1 矩阵 $A^{t,\beta}(c)$ 记录存储区域在时间点 t 新到达的 c -PAR 分配情况:矩阵元素 $a(c)_{rb}^\beta$ 取值原则与 $a(c)_{rb}$ 类似。为满足新到达的 PAR,需对存储区域进行垃圾回收处理,即对部分需擦除的数据块进行有效 PAR 迁移,然后擦除回收数据块。为此,采用 $m_c \times B$ 0-1 矩阵 $A^{t,\alpha}(c)$ 记录存储区域垃圾回收处理后现有 c -PAR 的分配情况:矩阵元素 $a(c)_{rb}^\alpha$



取值原则与 $a(c)_{rb}$ 类似。 $a(c)_{rb}^\alpha$ 与 $a(c)_{rb}$ 的值可能相同也可能不同, 当 $a(c)_{rb}^\alpha$ 与 $a(c)_{rb}$ 的值不同时, $a(c)_{rb}^\alpha$ 值为 1 代表存储第 r 个 PAR 的存储页面发生了改变。迁移前第 r 个 PAR 所在的存储空间肯定会被擦除回收, 否则失去了迁移的意义, 因此无需对第 r 个 PAR 迁移前的空间做失效标记处理。采用 B 维 0-1 向量 $E^{t,\alpha}$ 记录存储区域现有数据块, 是否进行了块擦除操作, 其元素 $e_b^\alpha=1$ 的取值表示对数据块 b 进行了擦除操作, 否则 $e_b^\alpha=0$ 。对于被擦除的数据块 b , 该块中的垃圾页面被回收, 对应的 $a(c)_{rb}^\alpha$ 被设置为 0。相应地, 计算有效数据迁移效用支出如下:

$$\kappa = \sum_{c=1}^C \sum_{r=1}^{m_c} \sum_{b=1}^B [l_m \cdot s_c \cdot a(c)_{rb} \cdot (1 - a(c)_{rb}^\alpha)] \quad (2)$$

其中, l_m 为有效数据迁移效用支出系数。计算数据块擦除效用支出如下:

$$\lambda = l_e \cdot \sum_{b=1}^B e_b^\alpha \quad (3)$$

其中, l_e 为数据块擦除效用支出系数。计算满足新到达的 PAR 效用收入:

$$\eta = \sum_{c=1}^C u_c \cdot \sum_{r=1}^{m_c} \sum_{b=1}^B a(c)_{rb}^\beta \quad (4)$$

其中, u_c 为 c-PAR 效用收入因子。擦除数据块回收了一些垃圾页面, 计算垃圾页面回收效用收入为:

$$\tau = \sum_{c=1}^C v_c \cdot \sum_{b=1}^B [e_b^\alpha \cdot \sum_{r=1}^{m_c} [a(c)_{rb} \cdot d(c)_r]] \quad (5)$$

其中, v_c 为垃圾 c-PAR 垃圾页面回收效用收入因子。对现有的 c-PAR 分配应遵循如下约束:

$$\sum_{b=1}^B a(c)_{rb} = 1, \forall r \in [1, m_c], \forall c \in [1, C] \quad (6)$$

$$\sum_{c=1}^C s_c \cdot \sum_{r=1}^{m_c} a(c)_{rb} \leq S, \forall b \in [1, B] \quad (7)$$

对新到达的 PAR 分配和垃圾回收应遵循如下约束:

$$1 - d(c)_r \leq \sum_{b=1}^B a(c)_{rb}^\alpha \leq 1, \forall r \in [1, m_c], \forall c \in [1, C] \quad (8)$$

$$e_b^\alpha \cdot a(c)_{rb} \cdot a(c)_{rb}^\alpha = 0, \forall r \in [1, m_c], \forall b \in [1, B], \forall c \in [1, C] \quad (9)$$

$$\sum_{b=1}^B a(c)_{rb}^\beta \leq 1, \forall r \in [1, m_c], \forall c \in [1, C] \quad (10)$$

$$\sum_{c=1}^C s_c \cdot [\sum_{r=1}^{m_c} a(c)_{rb}^\alpha + \sum_{r=1}^{n_c} a(c)_{rb}^\alpha] \leq S, \forall b \in [1, B] \quad (11)$$

$$d(c)_r \cdot a(c)_{rb}^\alpha \leq a(c)_{rb}, \forall r \in [1, m_c], \forall c \in [1, C] \quad (12)$$

式 (6)、式 (8) 保证每个 PAR 在同一个块分配页面; 式 (7)、式 (11) 表示单个数据块可容纳的 PAR 受限于数据块大小; 式 (8) 保证垃圾回收后, 有效数据仍然存在; 式 (9) 保证被擦除数据块的有效数据被迁移; 式 (10) 表示新到达的 PAR 可能被阻塞; 式 (12) 保证垃圾页面不会被迁移; 式 (9)、式 (12) 保证数据块被擦除后, 垃圾页面被回收。则考虑满足时刻 t 新到达 PAR 的垃圾回收最佳分配问题可描述为:

$$\max(\eta + \tau - \kappa - \lambda) \quad (13)$$

3.3 基于预测的先应式垃圾回收

在给定长度的时间段 $[t, t']$ 里, 新到达 c-PAR 数量 Y 总是有界的, 假设 Y' 为其一个上界。如果对时间段 $[t, t']$, 在时刻 t , 已经为 Y' 个 c-PAR 准备了可写存储空间, 则在时刻 t' 之前, 无需进行垃圾回收即可满足存储需求。基于存储资源的有限性, 在时刻 t 为 Y' 个 c-PAR 准备可写存储空间往往难以做到。可以降低 Y 的数值, 通过对 Y 基于置信水平 l 进行估计为 $Y(l)$, 保证大多数时候, 在时刻 t 为 $Y(l)$ 个 c-PAR 准备可写存储空间可以满足大多数情况下的系统需求。在实际工程应用中, 效用收益边际递减原理往往是普遍规律, 即在 Y 的估计值 $Y(l)$ 达到一定数量后, 通过增加 l 的值增加 $Y(l)$ 需要付出的代价往往得不偿失。因此, 为 Y 定义一个上位估计 $\bar{Y}$ 和下位估计 $\underline{Y}$ 是有意义的, 必须满足为 $\underline{Y}$ 个 c-PAR 准备可写存储空间, 而极大地满足为 $\bar{Y}$ 个 c-PAR 准备可写存储空间。

基于预测的先应式垃圾回收问题为: 假设在

时刻 t , 新到达的 **c-PAR** 数量为与 t 无关的随机变量 $X_{c,t}$, 在置信水平 l_c 的条件下, 给出未来一段时间直到时刻 t' 新到达的 **c-PAR** 总数量 Y_c 上限的一个下位估计 $\underline{n}'_c$ 和一个上位估计 $\overline{n}'_c$, 满足:

$$P([Y_c = \sum_{i=t}^{t'} X_{c,i}] \leq \underline{n}'_c) \geq l_c \quad (14)$$

$$P([Y_c = \sum_{i=t}^{t'} X_{c,i}] \leq \overline{n}'_c) \geq \bar{l}_c \quad (15)$$

其中, $\underline{l}_c$ 为将在时间点 t' 到达的 **c-PAR** 数量上限估计的下位置信水平, $\bar{l}_c$ 为将在时间点 t' 到达的 **c-PAR** 数量上限估计的上位置信水平, 显然 $\underline{l}_c \leq \bar{l}_c$ 。

根据大数定理, 容易知道 Y_c 服从正态分布 $N(\mu, \sigma)$, 可以对参数 μ 和 σ 作无偏估计。假设 $\varphi(x, \mu, \sigma)$ 为正态分布 Y_c 的概率密度函数, 则可以取 $\underline{n}'_c$ 和 $\overline{n}'_c$ 为:

$$\underline{n}'_c = \min\{y \mid \int_{-\infty}^y \varphi(x, \mu, \sigma) dx \geq \underline{l}_c\} \quad (16)$$

$$\overline{n}'_c = \min\{y \mid \int_{-\infty}^y \varphi(x, \mu, \sigma) dx \geq \bar{l}_c\} \quad (17)$$

由于有效数据迁移效用支出、垃圾页面回收效用收入只与现有 **c-PAR** 有关, 仍可根据式 (2)、式 (5) 计算有效数据迁移效用支出、垃圾页面回收效用收入。数据块擦除效用支出只与数据块有关, 仍可根据式 (3) 计算数据块擦除效用支出。给定集合 $\overline{\mathbf{n}}' = \{\overline{n}'_c \mid 1 \leq c \leq C\}$ 和 $\underline{\mathbf{n}}' = \{\underline{n}'_c \mid 1 \leq c \leq C\}$ 为未来一段时间新到达的 **PAR** 数量上限的上位预测和下位预测。相应地, 计算满足未来一段时间到达的 **PAR** 上位预测的效用收入:

$$\eta' = \sum_{c=1}^C u_c \cdot \sum_{r=1}^{\overline{n}'_c} \sum_{b=1}^B a(c)_{rb}^\beta \quad (18)$$

则基于上位预测的先应式垃圾回收问题可以描述为:

$$\max(\eta' + \tau - \kappa - \lambda) \quad (19)$$

通过最大化上位预测效用收益, 可以获取各类型 **c-PAR** 的预分配数量 $\mathbf{n}' = \{n'_c \mid 1 \leq c \leq C\}$, 对各类型 **c-PAR** 的预分配数量进行检测, 使其满

足下位预测要求:

$$\sum_{i=1}^c n'_i \geq \sum_{i=1}^c \underline{n}'_i, \forall 1 \leq c \leq C \quad (20)$$

当式 (20) 不满足时, 需要通过失效当前部分有效 **c-PAR**, 进行垃圾回收操作回收失效页面来满足未来需求。

4 算法描述

在时间点 t , PGCABP 算法主流程描述如下。

输入 $A^t = \{A^t(c) \mid 1 \leq c \leq C\}$,

$D^t = \{D^t(c) \mid 1 \leq c \leq C\}, \overline{\mathbf{n}}, \underline{\mathbf{n}}'$

输出 $(E^{t,\alpha}, A^{t,\alpha}, A^{t,\beta})$

(1) $(A^{t,\beta'}, \mathbf{n}') = \text{BSF}(A^t, \overline{\mathbf{n}}')$

(2) if $\sum_{i=1}^c n'_i = \sum_{i=1}^c \underline{n}'_i$ then return $(0, 0, A^{t,\beta'})$,

(3) $E^{t,\alpha}[0,0]=0, A^{t,\alpha}[0,0]=A^t, A^{t,\beta}[0,0]=A^{t,\beta'}, k_0^* = 0$

(4) for $c=1, 2, \dots, C$ do

if $n'_c < \underline{n}'_c$ then

$(E^{t,\alpha}[c, k_c^*], A^{t,\alpha}[c, k_c^*], A^{t,\beta}[c, k_c^*]) = \text{MaxUGC}(E^{t,\alpha}[c-1, k_{c-1}^*], A^{t,\alpha}[c-1, k_{c-1}^*], A^{t,\beta}[c-1, k_{c-1}^*])$

else

$k_c^* = 0; (E^{t,\alpha}[c, k_c^*], A^{t,\alpha}[c, k_c^*], A^{t,\beta}[c, k_c^*]) = (E^{t,\alpha}[c-1, k_{c-1}^*], A^{t,\alpha}[c-1, k_{c-1}^*], A^{t,\beta}[c-1, k_{c-1}^*])$

$n'_c = n'_c + k_c^*$

(5) for $c=1, 2, \dots, C$ do

if $\sum_{i=1}^c n'_i < \sum_{i=1}^c \underline{n}'_i$ then

$(E^{t,\alpha}, A^{t,\alpha}, A^{t,\beta}) = \text{IAGC}(E^{t,\alpha}, A^{t,\alpha}, A^{t,\beta})$

end

算法中第 (1) 步表示在不做有效数据迁移的前提下, 根据最大尺度优先原则, 依次考察可以满足分配的不同类型 **PAR** 数量; 如第 (2) 步所示, 如



果各类型 PAR 可分配的数量满足下位预测,则返回;否则,第(4)步根据最大垃圾回收效用原则,通过有效数据迁移,尽量满足需要分配的 PAR 数量;最后,如第(5)步所述,从大到小依次检查各种类型的 PAR 累积数量,当某种类型的 PAR 数量累积不超过下位预测时,需要通过释放该类型的有效 PAR 所占的空间,以便容纳未来将达到的 PAR。进一步说明如下。

(1)最大尺度请求优先(bigger size first, BSF)子算法

从要求分配最多页面的 PAR 开始,依次尝试在未来时间点 t' 前到达的 PAR 分配页面。得到临时分配矩阵 $A^{t,\beta'}$ 和分配向量 $\mathbf{n}' = \{n'_c | 1 \leq c \leq C\}$, $0 \leq n'_c \leq \overline{n'_c}$, 其中 n'_c 为采用 BSF 方法可完成分配的 c-PAR 数量。可以计算相应的 PAR 效用收入、有效数据迁移效用支出、数据块擦除效用支出、垃圾页面回收效用收入如下:

$$\eta_{\text{BSF}} = \sum_{c=1}^C u_c \cdot n'_c \quad (21)$$

$$\kappa_{\text{BSF}} = \lambda_{\text{BSF}} = \tau_{\text{BSF}} = 0 \quad (22)$$

(2)最大垃圾回收效用(maximum utility garbage collection, MaxUGC)子算法

如果对于所有的 c , 有 $n'_c = \overline{n'_c}$, 则算法终止。否则,从满足条件 $n'_c < \overline{n'_c}$ 的最小 PAR 类型索引序号 c 开始: 根据上次迭代所取得的当前数据块擦除向量 $E^{t,\alpha} [c-1, k_{c-1}^*]$, 新到达已分配 PAR 最佳分配矩阵 $A^{t,\beta} [c-1, k_{c-1}^*]$, 有效数据迁移矩阵 $A^{t,\alpha} [c-1, k_{c-1}^*]$, 通过垃圾回收整理以及再次满足 k_c^* 个 c-PAR 页面分配需求, 尝试得到最大效用净收益及本次迭代的最佳分配矩阵。边际效用利润示意图如图 3 所示。

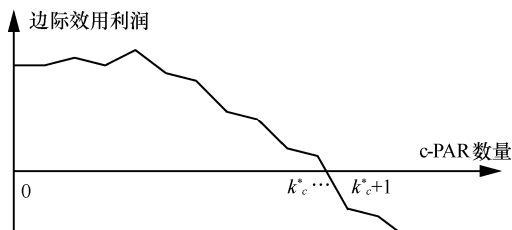


图 3 边际效用利润递减

如图 3 所示, 执行 BSF 子算法后, 如果 $n'_c < \overline{n'_c}$, 为继续满足 c-PAR, 需要进行垃圾回收整理, 垃圾回收将产生有效数据迁移效用支出、数据块擦除效用支出成本。随着满足分配的 c-PAR 数量的增加, 成本呈递增趋势, 增加一个 c-PAR, 需要付出的有效数据迁移效用支出、数据块擦除效用支出将越来越大, 即满足边际效用利润递减规律。因此, 采用 MaxUGC 子算法为特定类型的 c-PAR 寻求满足最大效用净收益的 k_c^* 值。由于 c-PAR 数量为整数, 所以不一定存在满足边际效用为 0 的 c-PAR 值。

假设执行 BSF 子算法后, 通过垃圾回收整理, 后续继续为 k ($0 \leq k \leq \overline{n'_c} - n'_c$) 个 c-PAR 分配了空间, 采用 $E^{t,\alpha} [c, k]$ 记录数据块擦除情况, $A^{t,\alpha} [c, k]$ 记录有效数据迁移情况, $A^{t,\beta} [c, k]$ 记录 PAR 放置情况, 则计算 PAR 效用收入 $\eta[c, k]$, 垃圾页面回收效用收入 $\tau[c, k]$, 有效数据迁移效用支出 $\kappa[c, k]$, 数据块擦除效用支出 $\lambda[c, k]$:

$$\eta[c, k] = \eta[c-1, k_{c-1}^*] + u_c \cdot k \quad (23)$$

$$\tau[c, k] = \sum_{c=1}^C v_c \cdot \sum_{b=1}^B [e[c, k]_b^\alpha \cdot \sum_{r=1}^{m_c} [a(c)_{rb} \cdot d(c)_r]] \quad (24)$$

$$\kappa[c, k] = \sum_{c=1}^C \sum_{r=1}^{m_c} \sum_{b=1}^B [l_m \cdot s_c \cdot a(c)_{rb} \cdot (1 - a(c, k)_{rb}^\alpha)] \quad (25)$$

$$\lambda[c, k] = l_e \cdot \sum_{b=1}^B e[c, k]_b^\alpha \quad (26)$$

其中, $a(c)_{rb}$ 为 $A'(c)$ 元素, $a(c, k)_{rb}^\alpha$ 为 $A^{t,\alpha} [c, k]$ 中 c-PAR 分配矩阵元素, $e[c, k]_b^\alpha$ 为 $E^{t,\alpha} [c, k]$ 元素, $d(c)_r$ 为 $D'(c)$ 元素。记 k_c^* 为对应最大效用净收益的 k 值。为获取 k_c^* , 可以通过连续调用最小垃圾回收成本(minimum cost garbage collection, MinCGC)子算法, 实现对新增 c-PAR 的分配, 直到完成 $\overline{n'_c} \sim n'_c$ 个 c-PAR 分配, 或者存储区域可用资源不够充足。此时, k_c^* 为连续调用 MinCGC 子算法操作中, 满足最大效用净收益(边际效用大于零)的次数。

MinCGC 子算法: 执行 BSF 算法后, 寻找参数 c^* , 使得对 $C \geq c > c^*$, 均有 $n'_c = \overline{n'_c}$ 个 c-PAR 已经分配, 且 $n'_c < \overline{n'_c}$ 。因此, 对于任意的 b ($1 \leq b \leq B$), 有 $s_{c^*} > W_b$ 。

确定参数 c^* 后, 初始化迁入和迁出两类数据块集合: (1) H_s 存放可进行垃圾回收的数据块; (2) H_d 存放具有可写页面的数据块。引入垃圾回收控制, 即仅满足 $c > c^*$ 类型的有效数据被迁移到目标数据块, 从而有:

$$H_s = \{b | G_b > 0\} \cup \{b | \exists c - \text{PAR}, c^* < c < C\} \quad (27)$$

$$H_d = \{b | W_b \in [s_c, s_{c^*})\} \quad (28)$$

为满足一个新到 c-PAR, 在 H_s 中确定一个数据块, 通过迁移其有效数据, 回收垃圾页面, 腾出更多空间。先确定 H_s 中具有最少有效页面的数据块 b^* , 如果 b^* 属于 H_d , 则 $H_d = H_d - \{b^*\}$ 。其次判断数据块 b^* 能否通过垃圾回收, 满足分配一个新的 c-PAR:

(1) 计算数据块 b^* 中 c' -PAR 数量 $m_{c'}(b^*)$, $c^* \leq c' \leq C$;

(2) 采用 BSF 计算 H_d 内可容纳的 c' -PAR 数量 $m_{c'}(H_d)$, $c^* \leq c' \leq C$;

(3) 确定可以从数据块 b^* 迁移的类型为 c' 的 PAR 数量 $m_{c'} = \min(m_{c'}(b^*), m_{c'}(H_d))$, $c^* \leq c' \leq C$;

(4) 确定满足 c-PAR 分配的条件为:

$$\sum_{c'=c^*+1}^C (m_{c'} \cdot s_{c'}) + W_{b^*} + G_{b^*} \geq s_c \quad (29)$$

对于满足式 (29) 的数据块, 采用首次命中 (first-fit, FF) 方法将数据块 b^* 中的有效 c-PAR 迁出; 否则 $H_s = H_s - \{b^*\}$, 循环至 H_s 为空。

(2) 迁移中转数据块

为保证数据块总能迁出有效数据进行回收, 定义一个迁移中转数据块 b_m , 在 BSF 算法中不对 b_m 进行 c-PAR 放置, 当迁移数据块 b 时, 可将 b 的有效数据迁入数据块 b_m , 然后更新迁移中转数据块 b_m 为 b 。

(3) 下位检测

为保证预分配的 c-PAR 满足下位预测要求,

采用失效并垃圾回收 (invalid and garbage collection, IAGC) 子算法, 使用式 (1) 所述 AERWH 算法进行热度评估, 释放一部分有效 c-PAR 所占的空间并进行回收, 然后实现预分配, 以满足式 (20) 所述下位预测条件。

(4) 算法复杂度

PGCABP 算法时间复杂度主要取决于运行 FF 子方法的次数, 运行一次 FF 子方法的时间复杂度为 $C^2 \cdot B$ 。在 MinCGC 子算法中最多调用 B 次 FF 子方法, 在 MaxUGC 子算法中最多调用 n (为预分配的 PAR 总量) 次 MinCGC 子方法。所以, PGCABP 算法时间复杂度约为 $O(n \cdot C^2 \cdot B^2)$ 。

(5) 算法适应性分析

通过不同的参数设置, 结合不同的预测条件, PGCABP 算法可以适用到不同的应用场景。

当未来一段时间到达的 PAR 下位预测为 0 时, 对应的应用场景为普通存储场景, 即 SSD 磁盘被作为存储使用, 不会因为新到达的 PAR 使已存储的 PAR 失效, 通过最大化效用收益, 期望通过垃圾回收整理, 可以存储更多的有效内容。

当考虑设置未来一段时间到达的 PAR 下位预测不为 0 时, 对应的应用场景为分级存储场景, 即 SSD 磁盘被作为缓存使用, 可以配合更便宜的存储如机械硬盘, 构成多级存储系统, 通过最大化效用收益, 期望 SSD 缓存更多的热点数据。

当考虑 $S_1 = S$, 且 $c > 1$ 时, 预测未来一段时间到达的 c-PAR 数量为 0 时, 对应的应用场景为 SSD 磁盘碎片整理, 即通过垃圾回收, 最大化效用收益, 期望获取最多的所含页面均为可写页面的空白数据块。

5 实验及应用

本文实验测试采用分布式文件系统 ZXDFS 实施, 搭建包括客户端、元数据服务器、共享存



表 1 测试应用案例

应用	应用来源	访问特点	文件大小
场景 1	机顶盒	顺序读	1~16 MB
场景 2	彩铃库	读多写少	64~1 MB
场景 3	邮箱	少数文件访问热度高	4~32 KB
场景 4	移动终端地理位置采集	顺序写	4 KB
场景 5	用户互联网行为分析	连续读, 随机写	128 byte~4 KB
场景 6	话单生成与计费	随机写, 随机读	32 byte

储设备节点的分级存储集群环境。各节点配置为: CPU 英特尔至强 E5606, 2.13 GHz 双路 16 核 64 位、内存 8 GB×7, 220 GB SSD、万兆网卡、操作系统 Susellsp3 V3.0.76-0.11。采用国内某运营商数据中心数据, 见表 1, 包含 6 种类型的典型应用场景。

实验的内容包括: 对每种类型的应用场景, 在相同的测试预置条件下, 对比 ZXDFS 的性能、成本在优化前和优化后的表现。优化前是指 ZXDFS 采用 NVMe 接口, 优化后是指 ZXDFS 通过 Open Channel 采用本文先应式垃圾回收算法。采用时间延迟作为性能指标进行评价, 采用 SSD 实际写入量作为 SSD 的寿命依据和成本指标进行评价。为保证试验结果可靠, 每种类型应用场景重复试验 10 次, 去掉最小的 2 个值, 去掉最大的 2 个值, 对中间的 6 个值取平均。

5.1 性能对比测试

在测试中, 采用文件读写访问时间延迟作为指标进行评价。优化后采用的是本文所述的先应式垃圾回收技术应用到分级存储系统。

性能对比测试结果如图 4 所示, 在测试的 6 种应用场景中, 场景 3 性能提升的效果最佳, 达到 54%。场景 1 性能提升的效果最不明显, 仅为 3%。其他场景性能提升比例介于 27%~37%, 与各场景的平均性能提升水平 32%较接近。场景 1 性能提升不显著, 是由于文件普遍较大, 同时主

要是顺序读写, 产生的碎片少, 优化难以发挥效果。场景 3 性能提升较大的原因是文件访问粒度小, 具有很强的局部性, 产生的碎片较多, 优化效果得以充分发挥。

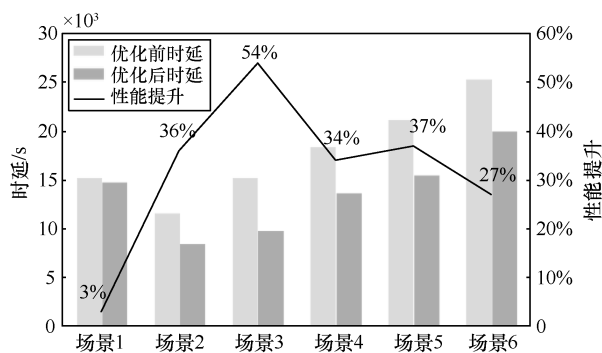


图 4 性能对比测试结果

5.2 成本对比测试

在测试中, 采用 SSD 实际写入量作为 SSD 寿命的判断依据, 在写入既定有效数据的条件下, SSD 实际写入量低意味着更长的使用寿命和更低的使用成本。SSD 的写入量可以通过“iostat”命令获取。

成本对比测试结果如图 5 所示, 在测试的 6 种应用场景中, 场景 6 降低成本的效果最明显, 达到了 80%。场景 4 降低成本效果最不明显, 仅为 6%; 其余应用场景降低效果为 26%~42%, 在所有场景降低成本比例的均值 36.5%左右波动。场景 6 降低成本效果明显的原因是垃圾页面碎片数量较大, 优化后避免了有效数据迁移

和有效数据二次搬迁问题,也就大大减少了写入数据量。场景4降低成本效果差是因为纯顺序写入,难以出现垃圾页面碎片,本文优化难以发挥效果。

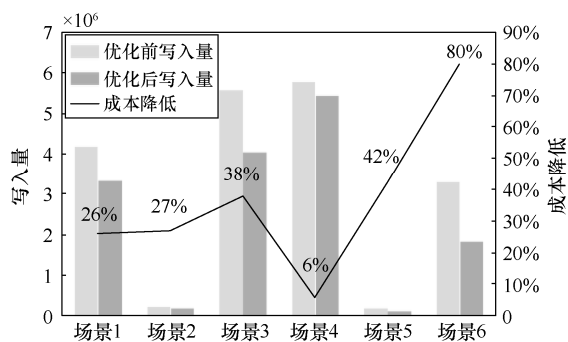


图5 成本对比测试结果

6 结束语

SSD 的存储特性和数据随机写会导致大量垃圾页面产生,而垃圾回收算法的优劣影响到 SSD 的性能和寿命。本文先介绍了 SSD 优化的相关研究和分析,总结了现有的垃圾回收方法及不足,提出了 PGCABP 算法。根据 AERWH 算法评估数据热度,实施分类存储区域的架构,并归一成某一时段最小化垃圾回收成本的优化问题。采用上位、下位两级门限对未来 PAR 数量进行预测,在满足未来 PAR 数量下位门限的同时,最大化满足未来 PAR 数量上位门限。阐述其数学定义和获取近似最优解的算法。PGCABP 算法在产品 ZXDFS 中的实际应用,验证了本文算法可以提升 SSD 的性能和使用寿命。但 PGCABP 算法还有改进之处,在下一阶段的研究工作中,将考虑结合具体业务场景下 PAR 的业务特性对 WA 问题进一步优化。此外,如何结合业务,和 SSD 厂商一起探索对现有预留空间、磨损均衡等技术进行深度优化,也是未来研究的方向之一。

参考文献:

[1] 黄平. 基于固态硬盘特征的存储优化研究[D]. 武汉: 华中科技大学

大学, 2013: 31-35.

HUANG P. Research on storage optimization based on solid state disk characteristics[D]. Wuhan: Huazhong University of Science and Technology, 2013: 31-35.

[2] WU S Z, CHEN X X, MAO B. GC-RAIS: garbage collection aware and redundant array of independent SSDs[J]. Journal of Computer Research and Development, 2013, 50(1): 60-68.

[3] MAD Z, FENG J H, LI G L. A survey of address translation technologies for flash memories[J]. ACM Computing Surveys, 2014, 46(3): 238-276.

[4] GUPTA A, KIM Y, URGONKAR B. DFTL: a flash translation layer employing demand-based selective caching of page-level address mapping[J]. ACM Sigarch Computer Architecture News, 2009, 37(1): 229-240.

[5] HU X Y, ELEFTHERIOUS E, HAAS R, et al. Write amplification analysis in flash-based solid state drivers[C]//The 2nd Int Systems and Storage Conference, May 4-6, 2009, Haifa, Israel. New York: ACM Press, 2009: 10-19.

[6] 姚英彪, 杜晨杰, 王发宽. 一种基于分类策略的聚簇页级闪存转换层算法[J]. 计算机研究与发展, 2017, 54(1): 142-153.

YAO Y B, DU C J, WANG F K. A clustered lpage-level flash translation layer algorithm based on classification strategy[J]. Journal of Computer Research and Development, 2017, 54(1): 142-153.

[7] JIANG S, ZHANG L, YUAN X, et al. S-FTL: an efficient address translation for flash memory by exploiting spatial locality[C]//Mass Storage Systems and Technologies, May 23-27, 2011, CO, USA. New York: ACM Press, 2011: 1-12.

[8] 张琦, 王林章, 张天, 等. 一种优化的闪存地址映射方法[J]. 软件学报, 2014, 25(2): 314-325.

ZHANG Q, WANG L Z, ZHANG T, et al. Optimized address translation method for flash memory[J]. Journal of Software, 2014, 25(2): 314-325.

[9] LEE D, CHOI J, KIM J H, et al. LRFU: a spectrum of policies that subsumes the least recently used and least frequently used policies[J]. IEEE Transactions on Computers, 2001, 50(12): 1352-1361.

[10] MEGIDDO N, MODHA D S. ARC: a self-tuning, low overhead replacement cache[C]//Usenix Conference on File and Storage Technologies, Mar 31-Apr 2, 2003, San Francisco, CA, USA. New York: ACM Press, 2003: 115-130.

[11] JIANG S, ZHANG X. LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance[J]. ACM Sigmetrics Performance Evaluation Review, 2002, 30(1): 31-42.

[12] 周权彪, 张兴军, 董小社, 等. 基于缓存映射项重用距离的



闪存地址映射方法[J]. 计算机研究与发展, 2018, 55(5): 1065-1077.

ZHOU Q B, ZHANG X J, DONG X S, et al. FTL address mapping method based on mapping entry inter-reference recency[J]. Journal of Computer Research and Development, 2018, 55(5): 1065-1077.



黄震江 (1976-), 男, 中兴通讯股份有限公司高级工程师, 主要研究方向为云计算和大数据。

[作者简介]



屠雪真 (1998-), 女, 河南大学在读, 主要研究方向为云计算和大数据。



陈正光 (1976-), 男, 中兴通讯股份有限公司高级工程师, 主要研究方向为数据库及存储。