基于NAND闪存的高性能和可靠的PRAID-6

陈金忠[1],姚念民[2],蔡绍滨[1]
(1.哈尔滨工程大学计算机科学与技术学院,黑龙江哈尔滨 150000;2.大连理工大学计算机科学与技术学院,辽宁大连 116000))

摘 要: 基于固态硬盘的RAID-6阵列,在每次数据更新时,都需要计算和写入校验信息,降低了阵列的性能和缩短了固态硬盘的使用寿命,该论文提出了一种基于延迟写入校验信息策略的RAID-6,称为PRAID-6.在每次数据更新时,PRAID-6只计算部分校验信息,写入非易失性存储器P-Cache.在垃圾回收时,将部分校验信息与原校验信息合并,产生新的校验信息,写入固态硬盘.通过实验测试结果表明PRAID-6的响应时间比RAID-6减少了30%,比RAID-5减少了10%.垃圾回收开销比RAID-6减少了20%,比RAID-5减少了15%.

关键词: RAID-6;固态硬盘;垃圾回收;部分校验;非易失性存储器
中图分类号: TP333.35 **文献标识码:** A **文章编号:** 0372-2112 (2015)06-1211-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.06.026

High Performance and Reliable PRAID-6 Based on NAND

CHEN Jin-zhong[1], YAO Nian-min[2], CAI Shao-bin[1]
(1. *College of Computer Science and Technology, Harbin Engineering University, Harbin, Heilongjiang* 150000, *China*;
2. *College of Computer Science and Technology, Dalian Univerrsity of Technology, Dalian, Liaoning* 116000, *China*)

Abstract: Solid state drives (SSDs)-based RAID-6 calculates parity and writes it to disk array when data is updated, which decreases the performance of disk array and reduces the life time of SSDs. We propose a partial parity-based RAID-6 (PRAID-6), which uses a delayed parity updating scheme to reduce the number of parity updates. Whenever a page is updated, PRAID-6 generates partial parity data of the stripe and stores it on a special non-volatile memory (NVRAM) called parity cache (P-Cache). The new parity is computed by partial parity and old version of parity when garbage collection mechanism is invoked. Then the new parity of the stripe is flushed to SSD. Experimental results show PRAID-6 reduces response time by 30% and 10% compared to RAID-5 and RAID-6, respectively. PRAID-6 reduces garbage collection overhead by 20% and 15% compared to RAID-6 and RAID-5, respectively.

Key words: RAID-6; SSD; garbage collection; partial parity; NVRAM

1 引言

过去几十年,由于闪存的体积小、抗震性、低功耗等特点,使之广泛用于MP3,PMP和移动电话[1].近年来,随着闪存的价格下降,NAND闪存技术逐渐应用于个人PC和企业存储系统.与传统硬盘不同,基于NAND闪存的固态硬盘(SSD)采用电子技术,存取速度比传统硬盘(HDD)快.通常,固态硬盘采用多层单元(MLC)架构来增加存储系统的容量,使用纠错码(ECC)保证数据可靠性.MLC使用的是多位BCH或者是RS编码,但多位纠错码的译码时间长,增加了硬件开销.另一种是采用廉价磁盘冗余阵列(RAID)技术来构建大容量、高性能和可靠的存储系统.针对固态硬盘,文献[2]提出了一种基于SSD的RAID-4架构,它计算新的校验数据并写入非易失性存储器NVRAM.当此"条带"上所有数据都更新完成时,才将校验数据写入固态硬盘.这种策略并没有减少校验数据的计算开销,每次计算校验数据时都需要读取原校验信息和更新前的页面.文献[3]将阵列中的磁盘分配不均匀的校验信息,使它们以不同的速度老化,从而保证阵列的可靠性.但这种提高可靠性的策略是以牺牲吞吐量为代价的.文献[4]采用延迟写入校验策略,在系统空闲时,才将校验数据写入固态硬盘,减少了校验数据写入频率.然而,如果在校验数据没有写入磁盘时,数据发生错误,那么将不能及时的恢复数据.文献[5]提出了一种基于SSD的RAID-5架构体系,减少了校验信息写入开销.但如果两块磁盘同时失效,数据就会丢失.所以,为了更加有效的保证数据的可靠性,两个独立分布式校验方案的RAID-6技术被应用于存储系

收稿日期:2013-06-26 修回日期:2014-07-06;责任编辑:梅志强
基金项目:国家自然科学基金(No.61073047);哈尔滨科技人才专项基金(No.2012RFLXG023);中央高校科研业务费专项资金(No.HEUCFT)

统.基于固态硬盘的 RAID-6,在每次数据更新时,需要写入双重校验信息,降低了 RAID-6 的性能和缩短固态硬盘的使用寿命.因此,本文提出了一种延迟写入校验信息的 RAID-6 模型,即 PRAID-6.PRAID-6 是在数据更新时,只计算部分校验数据,将部分校验数据写入非易失性存储器 P-Cache.在垃圾回收时,计算新的校验数据,写入固态硬盘.PRAID-6 减少了校验信息的处理开销,延长了固态硬盘的寿命,提高了阵列的性能.

2 相关工作

2.1 NAND 闪存

NAND 包括两种不同的架构,单层单元(SLC)和多层单元(MLC).SLC 架构是在每个 Cell 中存储 1 位的信息.MLC 架构是在每个 Cell 中存储多位信息.MLC 的容量比 SLC 大,但 MLC 读操作时延是 SLC 的两倍,写操作时延是需 SLC 的 4 倍[6].为了避免更新存储页时必须擦除整个存储块,NAND 闪存引入闪存转换层(FTL),主要功能分为映射机制[7],垃圾回收机制和损耗均衡机制.

2.2 RAID-6 校验和恢复原理

RAID 技术是由加州大学伯克利分校提出,它利用多个小容量的磁盘来增加存储系统的容量,同时利用冗余数据来保证磁盘阵列的可靠性.RAID-6 的双重校验信息使用不同的算法,第一重校验使用 XOR 算法来计算.第二重校验的生成是基于 Galois 域上的乘法和加法运算[8].

2.3 基于 SSD 的 RAID-6 的不足

基于 SSD 的 RAID-6 是由多个 SSD 构成的,双重校验信息均匀的分布于固态硬盘.条带号由逻辑页面号计算,当数据页面更新时,RAID-6 需要执行以下步骤:

(1)从 NAND 闪存芯片读取旧的双重校验信息和旧的页面.

(2)计算新的双重校验信息.

(3)写入新的页面和新的双重校验信息.

假设固态硬盘一次读操作的时延为 C_r,写操作时延为 C_w,那么一次数据更新操作的总时延 $C_t = 3C_r + 3C_w$.校验数据处理则需要三次读取操作与两次写入操作,其时延 $C_p = 3C_r + 2C_w$.因此,基于固态硬盘的 RAID-6,每次数据更新时,需要频繁的读写操作,降低了 RAID-6 的性能和缩短了固态硬盘的使用寿命.

3 延迟写入校验信息策略

PRAID-6 在数据更新时,只计算部分校验数据,并存储到非易失性存储器 P-Cache.非易失性存储器(NVRAM)是一种断电后数据不会丢失的随机存储器,如 PRAM 和 MRAM[9].在垃圾回收时,将部分校验数据与原校验数据合并,计算新的校验数据,写入固态硬盘.合并的过程称为校验信息的“提交”.更新前的页面称为“失效页”,在垃圾回收时,这些失效的页面会被擦除以回收存储空间.在 PRAID-6 中,当磁盘发生故障时,这些失效的页可用于恢复数据,因此,将这类页面称为“半失效页”.在使用寿命方面,固态硬盘的块擦除次数大约为 10^5,而 MRAM 的块擦除次数达到了 10^{15}.因此,与固态硬盘相比,MRAM 的损耗可以忽略不计.本文采用的延迟写入校验信息策略,减少了固态硬盘写入的次数,从而降低了发生垃圾回收的概率.因此,减少了块的擦除次数和页面复制开销.对于读请求,如果是页面发生错误,可以利用 ECC 来纠错.如果某个磁盘发生故障,可以利用其它磁盘的数据页恢复.否则,不需要计算或读出完整的校验信息.因此,延迟写入校验信息策略并不会影响固态磁盘阵列的读操作.

3.1 PRAID-6 部分校验信息的生成

假设将数据页 D_i 更新为 D_i',D_i 对应的条带 S_j.部分校验信息生成分为以下 3 类:

(1)如果 P-Cache 中没有 S_j 对应的部分校验数据,即 $S_j \notin$ P-Cache.那么部分校验数据 $P_j' = D_i'$,$Q_j' = k_i \otimes D_i'$.此时不需要额外读操作.

(2)如果 S_j 对应的部分校验数据存在,但 D_i 不是 P_j' 和 Q_j' 关联的数据页面,即 $S_j \in$ P-Cache 并且 $D_i \notin H(P_j')$,$D_i \notin H(Q_j')$.此时,$P_j' = P_j' \oplus D_i'$,$Q_j' = Q_j' \oplus (k_i \otimes D_i')$,不需要额外读操作.

(3)如果 S_j 对应的部分校验数据存在,并且 D_i 也是 P_j' 和 Q_j' 关联的数据页,即 $S_j \in$ P-Cache,$D_i \in H(P_j')$,$D_i \in H(Q_j')$.则 $P_j' = P_j' \oplus D_i' \oplus D_i$,$Q_j' = Q_j' \oplus (k_i \otimes D_i') \oplus (k_i \otimes D_i)$,需要一次读取 D_i 操作.

假设每个 Flash 芯片包含 2 个数据块,每个数据块由 4 页组成,图 1 给出了更新数据页 D_0,D_1 和 D_1' 的过程.更新 D_0 时,P-Cache 没有与 S_0 相关的部分校验,符合类(1)情形,此时 $P_0' = D_0'$,$Q_0' = k_0 \otimes D_0'$.更新 D_1 时,S_0 部分校验数据存在,但 $D_1 \notin H(P_0')$,$D_1 \notin H(Q_0')$,符合类(2)情形.所以,部分校验数据 $P_0' = D_0' \oplus D_1'$,$Q_0' = (k_0 \otimes D_0') \oplus (k_1 \otimes D_1')$.当 D_1' 更新为 D_1'' 时,$S_0 \in$ P-Cache 并且数据页 $D_1' \in H(P_0')$,$D_1' \in H(Q_0')$,符合类(3)情形.因此,部分校验数据 $P_0' = P_0' \oplus D_1'' \oplus D_1'$,并且校验 $Q_0' = Q_0' \oplus (k_1 \otimes D_1'') \oplus (k_1 \otimes D_1')$.此时需要从 Chip3 读取 D_1',计算新的部分校验数据.

3.2 PRAID-6 部分校验提交过程

部分校验的提交是指将部分校验数据与原校验数据合并,产生新的校验数据的过程.分为两种情形:

(1)当部分校验关联的数据页数量小于 $N/2$ 时,即 $|H(P_0')| = |H(Q_0')| < N/2$,将部分校验数据,原校验

数据和关联页更新前的页合并.

(2)当部分校验关联的数据页数量大于等于 $N/2$ 时,即 $|H(P_0')| = |H(Q_0')| \geq N/2$. 将部分校验数据与没有更新的数据页合并得到新的校验数据. 图 2(a)给出了部分校验关联数据页数量 $|H(P_0')| = |H(Q_0')| < N/2$ 时,部分校验信息的提交过程. 为了计算新的校验数据,首先执行三次读操作,分别是读旧的校验数据 P_0, Q_0 和更新前的数据页 D_0. 然后执行两次写操作,将新的校验数据 P_0^{new} 和 Q_0^{new} 写入 RAID-6 阵列. 图 2(b)给出了当部分校验关联数据页的数量 $|H(P_0')| = |H(Q_0')| \geq N/2$ 时,部分校验的提交过程,首先执行一次读取 D_2 操作. 然后执行两次 P_0^{new} 和 Q_0^{new} 写入操作.

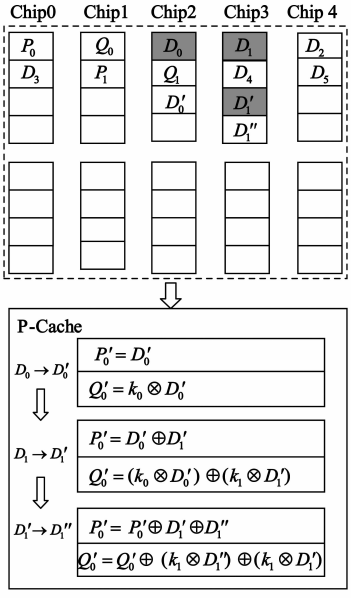


图1　RAID-6 部分校验生成过程

3.3　RAID6 与 PRAID-6 的时间开销分析

假设 $C_{\text{RAID-6}}$和 $C_{\text{PRAID-6}}$分别表示 RAID-6 和 PRAID-6 校验信息的开销. Num_p 表示更新数据页的数量. 由于传统 RAID-6 在每次数据页更新时,都需要三次读操作和两次写操作. 所以,$C_{\text{RAID-6}}$可表示为:

$$C_{\text{RAID-6}} = 3C_{\text{r}} \times \text{Num}_{\text{p}} + 2C_{\text{w}} \times \text{Num}_{\text{p}} \tag{1}$$

PRAID-6 首先将校验信息写入 P-Cache,然后提交到固态磁盘阵列. 因此,PRAID-6 的校验信息开销由写入 P-Cache 的开销和部分校验信息提交开销两部分构成. 分别记为 $C_{\text{wp-cache}}$和 $C_{\text{sub-p}}$. $C_{\text{PRAID-6}}$可表示为:

$$C_{\text{PRAID-6}} = C_{\text{wp-cache}} + C_{\text{sub-p}} \tag{2}$$

由于 P-Cache 的写入时延是 ns 数量级,固态硬盘的读取与写入时延是 μs 数量级,即 $C_{\text{wp-cache}} \ll C_{\text{sub-p}}$. 所以,$C_{\text{PRAID-6}} = C_{\text{sub-p}}$. 假设 PRAID-6 在更新 Num_{p} 数据页时,需要 q 个条带,每个条带关联的数据页数量为 $|H(P_i')|$($|H(P_i')| \geq 1, i = 1,2,\cdots,q$). 每个条带的部分校验提交开销为 C_{s_i}可表示为:

$$C_{s_i} = \begin{cases} (|H(P_i')| + 2)C_{\text{r}} + 2C_{\text{w}}, & |H(P_i')| < N/2 \\ (N - |H(P_i')|)C_{\text{r}} + 2C_{\text{w}}, & |H(P_i')| \geq N/2 \end{cases} \tag{3}$$

其中 N 表示 PRAID-6 磁盘的数量.

根据式(2)和式(3),可得到

$$C_{\text{PRAID-6}} = \sum_{i=1}^{q} C_{s_i} = \begin{cases} \text{Num}_{\text{p}} \times C_{\text{r}} + 2qC_{\text{r}} + 2qC_{\text{w}} & |H(P_i')| < N/2 \\ Nq\ C_{\text{r}} - \text{Num}_{\text{p}} \times C_{\text{r}} + 2qC_{\text{w}} & |H(P_i')| \geq N/2 \end{cases} \tag{4}$$

为了比较 RAID-6 与 PRAID-6 校验信息开销,计算 $C_{\text{RAID-6}}$与 $C_{\text{PRAID-6}}$的差值,记为 Δ,

$$\Delta = C_{\text{RAID-6}} - C_{\text{PRAID-6}} = \begin{cases} (2\text{Num}_{\text{p}} - 2q)C_{\text{r}} + (2\text{Num}_{\text{p}} - 2q)C_{\text{w}} & |H(P_i')| < N/2 \\ (4\text{Num}_{\text{p}} - Nq)C_{\text{r}} + (2\text{Num}_{\text{p}} - 2q)C_{\text{w}} & |H(P_i')| \geq N/2 \end{cases} \tag{5}$$

当 $|H(P_i')| \leqslant N/2$ 时,显然,条带数 $q \leqslant \text{Num}_{\text{p}}$. 所以,$\Delta \geq 0$. 在最坏的情形下,每个条带只关联一个数据页时,此时 $q = \text{Num}_{\text{p}}, \Delta = 0$.

当 $|H(P_i')| \geq N/2$ 时,显然条带数 $q \leqslant 2\text{Num}_{\text{p}}/N$. 此时,校验信息开销的差值 $\Delta \geq 2\text{Num}_{\text{p}}C_{\text{r}} + (2N - 4)\text{Num}_{\text{p}}C_{\text{w}}/N$.

因此,PRAID-6 减少了传统 RAID-6 带来的校验开销. 特别地,当每个条带关联的数据页都更新时,式(4)可表示为:

$$C_{\text{PRAID-6}} = 2\text{Num}_{\text{p}}C_{\text{w}}/N \tag{6}$$

在实际运行中,PRAID-6 对于监控录像采集,流媒体,Web Server Logging 和 SQL Server Logging 等顺序负载,可达到数量级的优化效果.

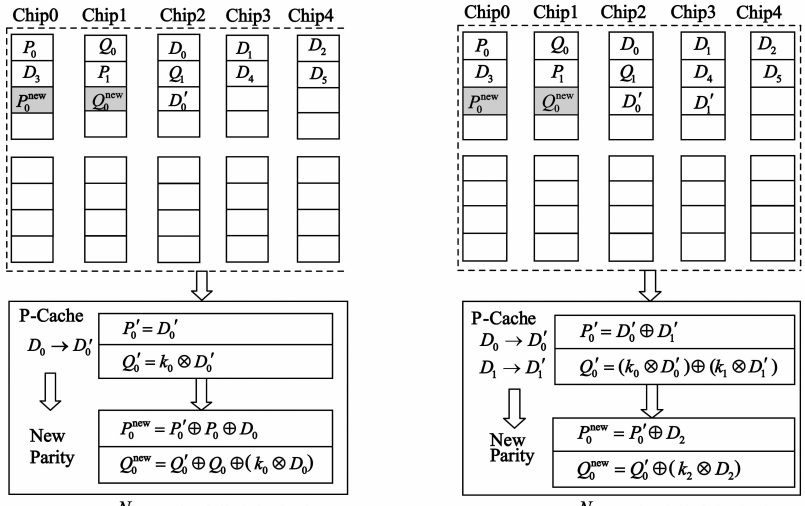
3.4　PRAID-6 垃圾回收机制

垃圾回收的开销可分为三类:全合并开销、部分合并开销和交换合并开销[10]. 其中,全合并需要大量的页面复制操作与块擦除操作. 部分合并开销只需要少量的页面复制操作与块擦除操作. 交换合并不需要页面复制操作,只需要一次擦除块操作. PRAID-6 在垃圾回收时,需要同时考虑页面复制的开销和提交部分校验信息的开销. 垃圾回收的开销由页面复制开销和部分校验信息的提交成本构成,即

$$\text{GC}_{\text{overhead}} = C_{\text{copy}}(\text{DB}) + C_{\text{commit}}(\text{DB}) \tag{7}$$

其中 C_{copy}(DB)表示回收块时页面复制的开销,C_{commit}(DB)表示校验信息提交的开销.

假设数据块的大小为 N_{p},有效页数量为 N_{v},则 C_{copy}(DB)可用下式表示:



(a) $|H(P'_0)|<\frac{N}{2}$ 部分校验信息提交过程　　(b) $|H(P'_0)|\geq\frac{N}{2}$ 部分校验信息提交过程

图2 PRAID-6部分校验提交过程

$$C_{\text{copy}}(\text{DB}) = N_{\text{v}} \times (C_{\text{r}} + C_{\text{w}}) \tag{8}$$

由 3.2 节可知,部分校验的提交成本 $C_{\text{commit}}(\text{DB})$可表示为:

$$C_{\text{commit}}(\text{DB}) = \sum_{i=1}^{(N_{\text{p}}-N_{\text{v}})} C_{s_i} \tag{9}$$

由式(8)和式(9),得到式(10):

$$\text{GC}_{\text{overhead}} = N_{\text{v}} \times (C_{\text{r}} + C_{\text{w}}) + \sum_{i=1}^{(N_{\text{p}}-N_{\text{v}})} C_{s_i} \tag{10}$$

PRAID-6 选取最小 $\text{GC}_{\text{overhead}}$的数据块回收.固态硬盘的垃圾回收是以块为单位,首先选择一个空闲块,然后将回收块的有效页复制到空闲块.最后,擦除回收块.PRAID-6 也采用上述垃圾回收过程.因此,这种垃圾回收方式不会造成数据碎片.

3.5 P-Cache 替换算法和 Destage 策略

当 P-Cache 没有空闲空间时,必须将一部分数据块刷新到固态硬盘阵列.经典的缓存替换算法诸如 LRU,LFU 等.然而,在本文所提出的 P-Cache 架构中,不仅要考虑替换算法的命中率,而且需要考虑部分校验提交成本.假设 $C_{\text{commit}}(\text{DB})$表示部分校验提交的成本,$U$ 表示数据块更新的频率.P-Cache 通过计算下式的值,并选取最小值 RP 对应的数据块 DB 进行替换.

$$\text{RT}(\text{DB}) = \alpha C_{\text{commit}}(\text{DB}) + (1-\alpha)U, 0 \leqslant \alpha \leqslant 1$$

考虑两种极端情况,α 为 1 时,完全按照校验数据的提交成本进行替换.α 为 0 时,完全按照 LRU 算法进行替换.对脏数据的刷盘操作,可通过 Destage 策略来实现,具体步骤如下:

(1)初始化系统所允许脏页数量的上限值和下限值,分别记为 $D_{\min}$和 $D_{\max}$.

(2)若当前脏页的数量 D_{cur}大于 $D_{\max}$,则进行刷盘操作,直到脏页数量小于 $D_{\min}$.

(3)若当前脏页数量 D_{cur}低于最小脏页数量,在系统空闲时刷新脏页.

(4)若脏页数量 D_{cur}介于 $D_{\min}$和 $D_{\max}$之间,在系统空闲时进行刷盘操作,直至脏页数量低于 $D_{\min}$.

3.6 PRAID-6 的数据恢复

基于 SSD 的 RAID-6 能够恢复两块磁盘同时出故障的情形.PRAID-6 数据恢复可分为以下四种情形:

情形 1:双重校验信息 P,Q 出错

情形 2:校验信息 P 与数据页 D 出错

情形 3:校验信息 Q 与数据页 D 出错

情形 4:两个数据页出错

其中 PRAID-6 的数据分布和 P-Cache 的内容如图 3 所示.恢复策略如下:

情形 1:

当 P_0,Q_0 出错,利用"半失效页" D_0,D_1 和数据页 D_2 重建校验数据,即

$$P_0 = D_0 \oplus D_1 \oplus D_2$$

$$Q_0 = (k_0 \otimes D_0) \oplus (k_1 \otimes D_1) \oplus (k_2 \otimes D_2)$$

情形 2 分为两类:

①当 P_0 与 D'_0 出错,$D'_0 \in H(P'_0)$并且 $D'_0 \in H(Q'_0)$.由 P'_0 恢复 D'_0 的计算开销比由 Q'_0 恢复 D'_0 小,所以 $D'_0 = P'_0 \oplus D'_1$.P_0 可由"半失效页" D_0,D_1 和数据页 D_2 重建,即 $P_0 = D_0 \oplus D_1 \oplus D_2$.

②当 P_0 与 D_2 出错,此时 $D_2 \notin H(P'_0)$并且 $D_2 \notin H(Q'_0)$,P_0 与 D_2 可由"半失效页" D_0,D_1 和校验数据 Q_0 恢复,数据页 $D_2 = k_2^{-1} \otimes [Q_0 \oplus (k_1 \otimes D_1) \oplus (k_0 \otimes D_0)]$,校验信息 $P_0 = D_0 \oplus D_1 \oplus D_2$.

情形 3 分为两类:

①当 Q_0 与 D'_0 出错,$D'_0 \in H(P'_0)$并且 $D'_0 \in H(Q'_0)$.

从而数据页 $D_0'=P_0'\oplus D_1'$. Q_0 可由“半失效页” D_0, D_1 和数据页 D_2 重新计算,即

$Q_0=(k_0\otimes D_0)\oplus(k_1\otimes D_1)\oplus(k_2\otimes D_2)$

②当 Q_0 与 D_2 出错, $D_2\notin H(P_0')$并且 $D_2\notin H(Q_0')$, Q_0 和 D_2 可由“半失效页” D_0, D_1 和校验数据 P_0 恢复,数据页 $D_2=P_0\oplus D_0\oplus D_1$,校验信息

$Q_0=(k_0\otimes D_0)\oplus(k_1\otimes D_1)\oplus(k_2\otimes D_2)$.

情形 4 分为三类:

①当数据页 D_0' 与 D_1' 出错, $D_0', D_1'\in H(P_0')$并且 $D_0', D_1'\in H(Q_0')$,可由 $P_0'=D_0'\oplus D_1'$ 和 $Q_0'=(k_0\otimes D_0')\oplus(k_1\otimes D_1')$联立恢复数据页 D_0' 和 D_1',得到数据页 $D_0'=Q_0'\oplus(k_1\otimes P_0')/(k_0\oplus k_1)$和 $D_1'=P_0'\oplus D_0'$.

②当 D_0' 和 D_2 出错,由于 $D_0'\in H(P_0')$并且 $D_0'\in H(Q_0')$,因此数据页 $D_0'=P_0'\oplus D_1'$. D_2 可由“半失效页” D_0, D_1 和 P_0 得到,即 $D_2=P_0\oplus D_0\oplus D_1$.

③当 D_1 和 D_2 出错, $D_1, D_2\notin H(P_0')$并且 $D_1, D_2\notin H(Q_0')$,此时 D_1 和 D_2 可由“半失效页” D_0,校验数据 P_0 和 Q_0 恢复.从而,恢复得到的数据页

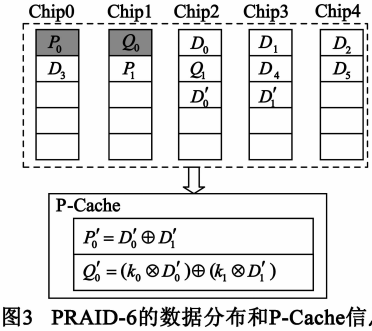


图3　PRAID-6的数据分布和P-Cache信息

4　性能测试

4.1　测试负载与性能指标

测试的负载包括 IOzone, Postmark 和 TPC-C, Financial1 和 Financial2. Postmark 负载的特点是频繁和大量的存取小文件. IOzone 是对大文件的读写性能测试. TPC-C 是产生数据库事务处理的负载.它是用来模拟数据库实时事务处理的基准程序. Financial1 是以写为主的负载,Financial2 是以读为主的负载性能指标包括阵列的平均响应时间校验开销和垃圾回收开销.

4.2　固态磁盘阵列的平均响应时间

图 4 给出了 RAID-0、RAID-5, RAID-6 和 PRAID-6 在 IOzone、Postmark 和 TPC-C 三种负载下的响应时间.由于 RAID-0 只将数据以条带的方式分布于 Flash 芯片,没有提供校验功能,在更新数据时不需要进行校验信息的计算与写入,所以 RAID-0 在三种负载下的响应时间比 RAID-5、RAID-6 和 PRAID-6 低. RAID-5 更新一页时,需要进行 2 次读取操作与 2 次写入操作,而 RAID-6 需要 3 次读取操作与 3 次写入操作,因此,RAID-5 的响应时间比 RAID-6 少. PRAID-6 的响应时间比 RAID-6 减少了 30% 左右,比 RAID-5 减少了 10% 左右.这是由于 PRAID-6 在每次更新数据时,只计算部分校验数据,写入非易失性存储器 P-Cache. 垃圾回收时,产生新的校验数据,写入态硬盘.

4.3　固态磁盘阵列的校验开销

平均写校验开销指的是写操作的平均校验开销. RAID-0 没有校验开销.图 5 显示了 RAID-5, RAID-6 和 PRAID-6 在 IOzone, postmark 和 TPC-C 负载下的平均写校验开销.其中,PRAID-6 平均写校验开销比 RAID-6 减少了 31%,比 RAID-5 减少了 15% .PRAID-6 采用延迟写入校验信息策略,减少了校验信息的计算和写入开销.

4.4　固态磁盘阵列的块擦除次数

PRAID-6 采用 3.4 节的垃圾回收策略,不仅考虑了垃圾回收过程中页面迁移开销,而且考虑了部分校验信息的提交开销.图 6 给出在 P-Cache 大小为 64KB 时的 RAID-5、RAID-6 和 PRAID-6 在三种负载下相对于 RAID-0 的块擦除次数. PRAID-6 块擦除数比 RAID-6 减少了 20.2%,比 RAID-5 减少了 7.6%.由于 PRAID-6 延迟写入校验数据,使得固态硬盘的空闲块数比 RAID-6 和 RAID-5 多,因此,减少了块擦除次数.表 1 给出了不同 P-Cache 容量下的 RAID-5、RAID-6 和 PRAID-6 相对于 RAID-0 的块擦除次数.可以看出,随着 P-Cache 容量的增加,PRAID-6 的块擦除次数明显减少.随着 P-Cache 容量增加,缓存命中率逐渐提高.因此,块擦除次数逐渐减少.由于 RAID-5 和 RAID-6 没有使用 P-Cache,因此,块擦除次数基本不变.

表 1　不同 P-Cache 容量下的 RAID-5, RAID-6, PRAID-6 相对 RAID-0 的块擦除次数

	RAID-5	RAID-6	PRAID-6
16KB	1.331	1.512	1.301
32KB	1.329	1.522	1.282
64KB	1.310	1.521	1.233
128KB	1.330	1.520	1.109

4.5　固态磁盘阵列的垃圾回收开销

垃圾回收开销包括块擦除开销和页面迁移开销.图 7 显示了 RAID-5、RAID-6 和 PRAID-6 在 IOzone、Postmark 和 TPC-C 三种负载下相对于 RAID-0 垃圾回收的开销.RAID-0 没有校验信息写入,垃圾回收开销最小. PRAID-6 垃圾回收开销比 RAID-6 减少了 20%,比 RAID-5 减少了 15%. PRAID-6 采用延迟写入校验信息的策略,并且在垃圾回收过程中,考虑了页面复制开销和校验信息的提交成本.

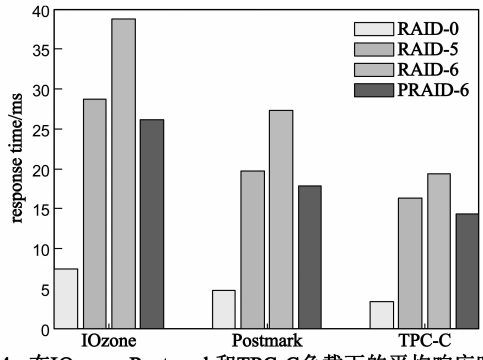


图4 在IOzone、Postmark和TPC-C负载下的平均响应时间

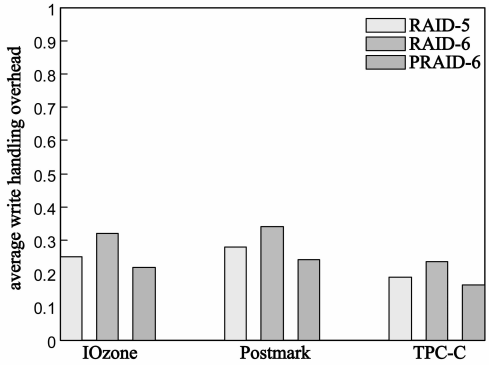


图5 IOzone、Postmark和TPC-C负载下的平均校验开销

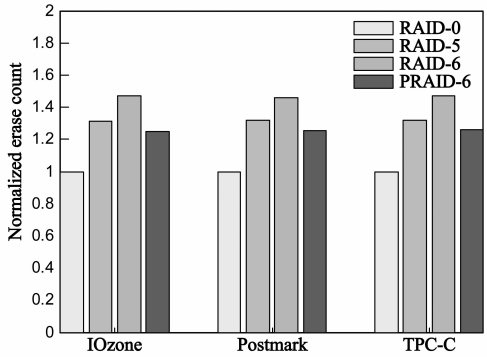


图6 RAID-5、RAID-6和PRAID-6相对RAID-0块擦除次数

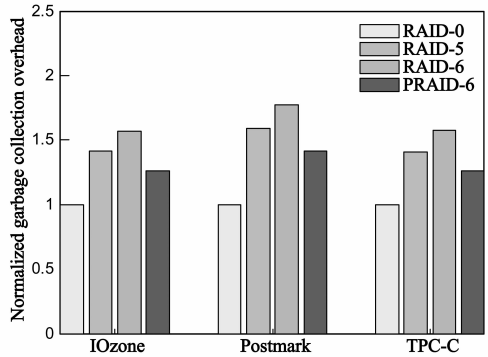


图7 RAID-5、RAID-6和PRAID-6相对于RAID-0 垃圾回收开销

4.6 P-Cache 相关测试

图 8 和 9 分别给出了不同 P-Cache 尺寸下的部分校验提交开销和命中率. 可以看出, Financial1 和 Financial2 负载随着 P-Cache 容量的增加, 其部分校验提交开销逐渐降低, 而 IOzone 负载的部分校验提交开销基本不变. 由于 Financial1 和 Financial2 都是随机访问负载, 具有局部性特征, 随着 P-Cache 容量的增加, 命中率逐渐提高. 因此, 部分校验提交开销逐渐降低. 而 IOzone 是顺序访问负载, 其部分校验提交开销和命中率基本不变. 另外, IOzone 负载具有顺序访问的特性, 因此, 其部分校验提交开销比 Financial1 和 Financial2 低. 对于顺序访问的负载, 固态硬盘阵列的校验信息提交成本是很低的, 这与 3.3 节的理论分析相吻合.

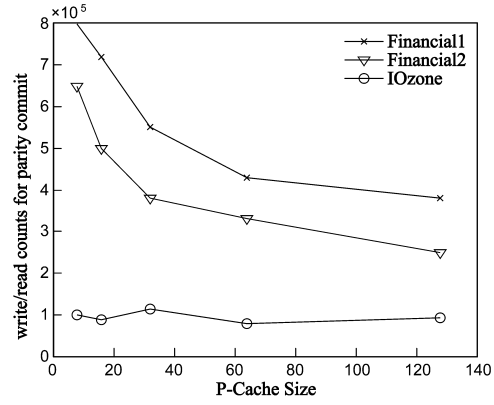


图8 部分校验提交开销与P-Cache容量关系

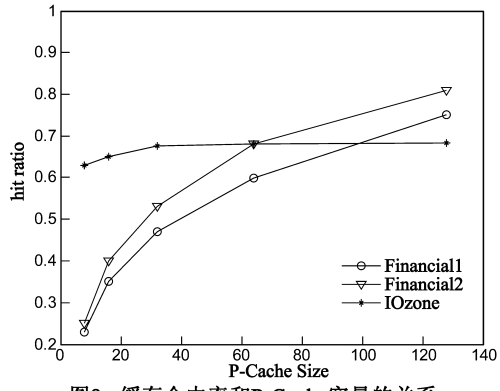


图9 缓存命中率和P-Cache容量的关系

5 结束语

本文提出了基于延迟写入校验信息策略的 RAID-6 模型, 即 PRAID-6. PRAID-6 在数据更新时, 计算部分校验数据并写入非易失性存储器 P-Cache, 利用“半失效页”恢复发生故障的数据. 在垃圾回收时, 计算新的校验数据并写入固态硬盘. PRAID-6 减少了垃圾回收的开销和延长了固态硬盘的使用寿命. 在未来的工作中, 将研究固态硬盘内部的 FTL 策略.

参考文献

[1] Gal E, TOLEDO S. Algorithms and data structures for flash

memories[J]. ACM Computing Surveys 37, 2005, 37(2): 138 – 163.

[2] K Greenan, D D E Long, E L Miller, T Schwarz, A Wildani. Building flexible, fault-tolerant flash-based storage systems [A]. Proc Fifth Workshop Hot Topics in System Dependability (HotDep '09)[C]. Estoril, Portugal: IEEE Press, 2009. 165 – 179.

[3] KADAV A, BALAKRISHNAN, M PRABHAKARAN, MALKHI D. Differential RAID: Rethinking RAID for SSD reliability[A]. Proceedings of the Workshop on Hot Topics in Storage and File Systems (HotStorage'09)[C]. Big Sky, Montana: IEEE Press, 2009.55 – 59.

[4] Y Lee, S Jung, Y H Song. FRA: A flash-aware redundancy array of flash storage devices [A]. Proc Seventh IEEE/ACM Conf Hardware/Software Code sign and System Synthesis (CODES + ISSS '09)[C]. Grenoble, France: IEEE Press, 2009. 163 – 172.

[5] S Im, D Shin. Flash-aware RAID techniques for dependable and high-performance flash memory SSD [J]. IEEE Trans Comput, 2011, 60(1): 80 – 92.

[6] N Agrawal, V Prabhakaran, T Wobber, J D Davis, M Manasse, R Panigrahy. Design Tradeoffs for SSD Performance[A]. Proc of USENIX Technical Conference[C]. Boston, MA, US: IEEE Press, 2008. 57 – 70.

[7] T Chung, D Park, S Park, D Lee, S Lee, H Song. System software for flash memory: a survey[A]. Procof ICEUC'06[C]. Seoul: Springer Berlin Heidelberg, 2006. 394 – 404

[8] 金超,冯丹,刘景宁,田磊. RAID6 编码的扩展算法及性能研究[J]. 电子学报, 2012, 40(1): 173 – 178.
JIN Chao, FENG Dan, LIU Jingnin, TIAN Lei. Code shortening and performance analysis for RAID6 codes[J]. Acta Electonica Sinica, 2012, 40(1): 173 – 178. (in Chinese)

[9] R F Freitas, W W Wilcke. "Storage-class memory: The next storage system technology[J]. IBM J. Research and Development, 2008, 52(4): 439 – 447.

[10] Gupta, A Kim, Y Urgaonkar B. DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings [J]. ACM SIGPLAN Notices, 2009, 44(3): 229 – 240.

作者简介

陈金忠　男,1986 出生,安徽合肥人,博士,哈尔滨工程大学博士生,主要研究方向为 Linux 存储系统,I/O 调度算法和固态硬盘,DSP 等.
E-mail: chenjin_zhong@126.com

姚念民　男,1974 出生,黑龙江大庆人,博士,大连理工大学计算机学院教授、博士生导师,主要研究方向为网络存储,信息安全,无线传感器网络以及高可信计算.
E-mail: lucos@dlut.edu.cn

蔡绍滨　男,1973 出生,黑龙江哈尔滨人,博士,哈尔滨工程大学计算机学院教授、博士生导师,主要研究方向为无线传感器网络,水声传感器网络.
E-mail: caishaobin@hrbeu.edu.cn