武汉大学 计算机学院

2017级硕士研究生课程

**海量存储技术结题报告**

学院： 计算机学院

学号： 2017282110230

姓名： 李超

任课老师： 何水兵

2017年12月

目录

1 作业组成 4

2 Last-level Cache Deduplication 4

2.1 摘要 4

2.2 该篇论文所作出的贡献 4

2.3 提出的动机 5

2.4 缓存重复数据删除的挑战 5

2.5 针对LLC的重复数据删除的新的设计 6

2.5.1 标签数组 7

2.5.2 数据数组 7

2.5.3 哈希表 7

2.6 操作 8

2.6.1缓存未命中 8

2.6.2 缓存命中 8

2.7 方法 9

2.7.1 仿真环境 9

2.7.2 基准 9

2.8 实验结果 10

2.8.1 性能优化 11

3 Extending Lifetime of SSD in RAID5 Systems through a Reliable Hierarchical Cache 13

3.1 论文背景与动机 13

3.2 RAID简介 13

3.2.1 RAID5系统介绍 13

3.2.2 SSD优点与缺点 14

3.2.3 RAID5中使用SSD的寿命问题 14

3.2.4 延长SSD寿命的方法 15

3.3 RH-Cache设计目标 16

3.4 RH-cache的总体架构 16

3.4.1 高级写缓存 17

3.4.2 低级写入缓存模块 19

3.4.3软件RAID控制器体系结构 19

3.5 硬件要求 20

3.5.1 硬件配置表 20

3.6 实验评估 21

3.7 总结 24

# 1 作业组成

本次作业由两篇论文组成，一篇来自于第二十八届ACM超级计算国际会议，标题为：**Last-level Cache Deduplication**.一篇来自于**2017年网络，架构和存储国际会议（NAS），标题为：Extending Lifetime of SSD in RAID5 Systems through a Reliable Hierarchical Cache .**

# 2 Last-level Cache Deduplication

最后一级缓存重复数据删除

## 2.1 摘要

高速缓存是现代微处理器性能的关键，近年来的研究工作专注于利用参考位置来提高缓存的效率。然而，价值冗余是另一个潜在的改进方法。本文作者发现基准程序的工作组中的许多块具有相同的值，因此，作者提出缓存重复数据删除来有效提高最后一层高速缓存的容量。之前的工作中，作者利用特定的值冗余压缩，现在计划检测重复的数据块并可通过多个物理地址访问的方式来只存储数据的一个副本。作者发现在特定的基准程序中存在着显著的值冗余，远远超出了在任何程序中零内容块块数的期望值。对于8MB的末级高速缓存，基于重复数据删除的方法在提高12%的存储容量的同时减少了12.2%的物理区域，其生产性能平均也提高了15.2%。

## 2.2 该篇论文所作出的贡献

（1）大多数的复制存在于高速缓存中，并量化缓存复制在18 SPEC CPU2006基准中的影响。

（2）提出了一个统一的缓存的重复数据删除技术，通过增加有效缓存容量来提高缓存性能。通过删除块级别的值冗余，缓存重复数据删除技术可显著提高缓存效率有限的面积和功耗。

（3）提出了一个新的末级缓存（LLC）设计和重复数据删除。相比于传统的末级缓存，重复数据删除的末级缓存使用大致相等的芯片面积和功耗，其性能却是传统的末级缓存的两倍。

基于作者所做的实验，缓存重复数据删除技术相比于基准8MB末级缓存，其性能提高了15.2%，这与传统的16MB末级高速缓存相当，并且优于12MB和14MB高速缓存，同时比8MB的末级缓存使用的物理区域要少12.2%。

## 2.3 提出的动机

传统的高速缓存设计浪费了容量，因为它存储重复的数据。当发出存储器请求时，从主存储器获取的数据也被带入用于未来请求的高速缓存中。该数据与从其物理存储器地址导出的标签相关联。 然而，具有不同块地址的高速缓存块可能包含相同的数据。相同的数据块在缓存中被复制是因为其标签不同。

通过测量在执行18个SPEC CPU2006基准期间存储在2MB 16路LLC中的不同块的百分比，每个基准运行10亿条指令的间隔。每隔一千万条指令计算不同块的数量。不同块的比率随工作负载而变化，但是对于所有基准，在高速缓存中始终存在重复的块。在基准中，hmmer具有最小的重复块百分比（平均为2.7％），zeusmp具有最大的重复块百分比（97.8％）。平均来说，在所有的基准测试中，35.1％的缓存块是被复制的。

大多数复制包含由输入和/或计算产生的非零数据值，其具有取决于程序行为的副本数目的随机分布的特征。作为示例，作者采用xalancbmk的随机执行点来显示重复度和重复数据的性质。 在随机执行点，在2MB高速缓存中，在29,278个高速缓存块中有14,931个不同块（即，51％块是不同的）。 有2,414个数据块与每个标签相关联，因此16％的块被复制一次。 有1,157个零内容块。 如果只压缩零内容块，则只保存总容量的4％。 如果可以从高速缓存中消除所有副本，则可以节省2MB高速缓存的38％以上的容量，这比现代处理器的典型256KB L2高速缓存大三倍。

## 2.4 缓存重复数据删除的挑战

重复数据删除是一项删除重复数据副本的特定操作。它已经被广泛应用于磁盘存储系统。重复数据删除后，相同的数据物理上只保存一份。冗余数据存储为相应的删除处理数据的引用，以提高存储利用率。尽管重复数据删除通常用于磁盘存储和内存压缩数据存储，但由于以下问题它在缓存上仍是一个挑战：

如何检测重复数据：第一个挑战是通过比较数据的方式来检测可能的重复。可以通过将分析数据与所有存储数据或特定部分存储数据进行比较来发现重复数据。

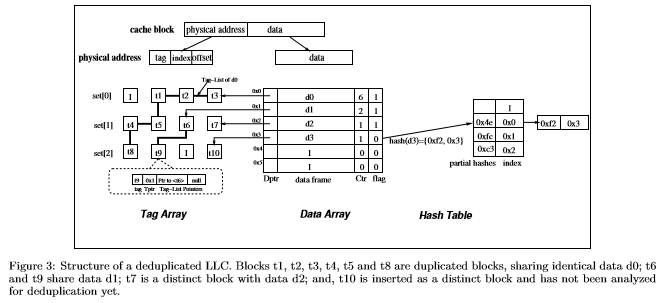
当检测到重复数据时：第二个挑战是针对重复数据检测的过程。缓存在弥合处理器和主存储器之间的性能差距发挥着重要作用，其中的访问延迟是系统整体性能的关键。重复数据删除的过程不应该影响缓存延迟。

重复数据删除的粒度：以前使用子块级别的粒度来压缩可能的可压缩数据。子块级别的粒度可能导致更高的删除率，但是它也导致访问延迟的增加和额外的开销。虽然子块级别的重复删除可以使删除更高效，但是系统性能可能会因为访问延迟的增加而退化。使用删除粒度来权衡删除程度和缓存延迟和开销是缓存重复数据删除的另一个挑战。

## 2.5 针对LLC的重复数据删除的新的设计

作者提出一个实用的LLC设计消除重复缓存块,称之为删除处理LLC。为了解决前一节中提到的挑战，删除处理LLC使用增广散列来检测重复，它速度快并且缓存容量利用率高。它使用后处理检测来隐藏可能增加的缓存延迟。它使用块级删除处理粒度，忽略其内容，通过分析块和缓存中已存储的数据对比，充分利用有限的开销来删除重复数据。对于重复块的替换策略，为了提高效率，作者提出了截然不同的随机替换（DFRR）策略。

如图，显示了删除处理LLC的结构。它由三个分离的部分:一个标签数组,一个数据数组,和一个哈希表。



### 2.5.1 标签数组

标记数组是一组保持缓存块的语义的关联结构。标记数组中的每个条目包含以下字段:传统的缓存中缓存块的元数据(例如标记位,LRU位,有效位,和修改位),一个索引数据数组的引用,和两个指向标记保存另一个双链接列表标签，其全部指向相同的数据块的引用。数据录入的引用,称为数据指针标记(Tptr),标识一个数据数组中的不同的条目。当有一个标签匹配,Tptr直接索引与这个缓存相关的数据块。当标记数组中插入一个标签时,它也是插入(如果有的话)与相应的数据有关的重复的块的双链接列表的标签。

当标签数组中的标签被替换时,它也从链表中删除。通过这些指针,所有标签存储在标签数组中，其中共享相同的有联系的数据。重复的块的链表的标签被称为标记列表和每个标签条目的两个指针被称为标记列表指针。当数据数组有替换时,所有相关的标签被删除或失效。标记数组只能被视为传统的缓存用来存储元数据。它使用请求的内存地址搜索匹配标签的具体设置。当缓存错误发生时,标记数组使用常规的缓存替换策略(即最近最少使用(LRU)而不是使用DFRR政策替代数据数组)。在实验中,为了公平在标记数组中我们使用传统的最近最少使用(LRU)替代策略。

### 2.5.2 数据数组

数据数组中的每个条目包含一个数据帧,一个计数器,一个指针,一个一比特重复数据删除标志。计数器(称为Ctr)表示的存储在标记数组的共享这些数据的标签数。当标签插入到标签数组,数组中相应的Ctr数据递增1。当标记数组中标签被替换或无效时,相应的Ctr减1。当Ctr为零,该数据块可以重用。指针(称为数据标记指针(Dptr))标识标记列表的头。无效条目的Dptr是用来保持一个空的可用数据条目的列表。一比特的重复数据删除标志指示是否为当前数据块进行了重复数据删除分析。数据数组可以被视为直接映射缓存，只可以被从相应的标签条目的Tptrs访问。

### 2.5.3 哈希表

删除处理LLC中第三个结构是一个增广哈希表。作者使用一个增广哈希表实现两层查找大部分缓存容量。第一级的查找发生在哈希表散列索引的数据,第二层次发生在通过存储哈希节点的数据索引的数组重定向。减少哈希碰撞的数量,哈希表实现为一系列像小桶的关联数组。一桶中每个节点包含一个16位指针数组索引数据,1位有效位,一个15位部分散列值。

## 2.6 操作

去重复的高速缓存在缓存命中和缓存未命中方面有不同的操作。在高速缓存访问过程中，所请求块的标签与特定标签阵列里的所有标签并行比较。如果查找失败，则发生缓存未命中，否则，发生缓存命中。

### 2.6.1缓存未命中

缓存未命中时，所请求的块被从主存中取出作为常规缓存。然后缓存块的存储空间会被分为两部分：标签阵列和数据阵列。数据块会被放置在一个从使用Dptrs维护的自由列表中随机选择的无效数据的条目中。标签块被相应地放置在使用内存地址索引的标签阵列中。此后标签的条目指针和数据的条目指针会进行更新指向彼此，同时命中率提高。如果标签阵列中没有无效条目，定期更替策略（LRU实验）将会选择一个更替的对象。如果数据阵列项没有无效条目，我们使用dfrr去选择一个更替对象。

此时，不对所请求的缓存块进行重复分析（撤销重复数据删除标志位）。相反，它直接使用未设置重复数据删除的标志位被直接放置在缓存中，表示它没有处理重复数据删除，并且不产生任何重复数据删除延迟。这个块的重复检测将不会被启动，除非下一个缓存未命中发生，如3.3节所述。故数据替换对象相应的哈希节点是无效的。

### 2.6.2 缓存命中

缓存命中可以是一个读命中或写命中。在一个重复数据删除缓存中，写命中会修改块中的数据，导致对更新的数据进行另一个重复检测的再散列，而读命中与重复数据删除无关，因此读命中和写命中操作是不同的：

当一次读操作命中了标签阵列，匹配项中的标签指针直接将数据阵列索引到检索所请求的数据。然后在标签阵列中实现复位信息更新。数据阵列不变。

当一次写操作命中了标签阵列，所请求的数据会被标签指针索引。如果它是一个独特块（Ctr为1），数据可以被立即更改，同时重复数据删除标识会被撤销，用来表明这是未被分析块。如果这是一个重复数据删除块，一个无效数据条目会被分配，用来放置更新数据，而不是直接更新数据阵列。在这种情况下，写命中到一个重复数据的处理和缓存未命中类似。此后标签条目的脏位和复位信息都会被更新。

## 2.7 方法

本节概述作者使用的实验方法

### 2.7.1 仿真环境

使用MARSSx86周期精确仿真器[26]，一个运行单核和多核工作负载的x86-64架构的全系统仿真，以评估提出的重复数据删除LLC。它模拟一个无序的4阶宽的x86处理器，其具有128条目重排序缓冲区以及具有MESI协议和片内互连的一致性高速缓存。微架构参数与Intel Core i7处理器一致，包括三级缓存层次结构：L1 I-cache和L1 D-cache，L2缓存和共享LLC。 L1和L2缓存对每个核是私有的。L1 I-cache和D-cache每个为4路32KB，L2缓存为8路256KB。共享LLC是一个统一的16路每核2MB的缓存。L1缓存，L2缓存，LLC和主存储器的访问延迟分别为4,10,40和250个周期。对于去重复的LLC，可以增加集合的数量和标签阵列的关联性以适应更多的块。我们分别通过将标签数组的集合数和关联性加倍来​​评估这两个想法。让标签阵列的尺寸加倍的原因是将重复的LLC与两倍的常规LLC进行比较。标签阵列的实际尺寸可以任意地增加，以在相应的功率和面积消耗方面实现更好的性能。基于实验，具有双倍标签阵列的评估的去重复LLC与基线的LLC的面积拟合。我们在第6节中展示了一个详细的成本分析。标签数组中的替换策略是LRU，而数据数组中的替换策略是提出的DFRR。

### 2.7.2 基准

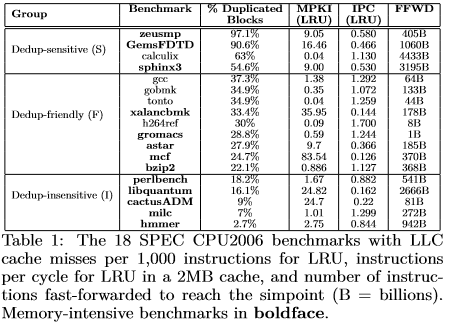
实验中使用的基准都是从规范CPU2006基准套件中随机选取。我们用SimPoint来识别单个基准的百万指令的时间间隙。每个基准都是用x86 - 64指令集来编译和运行由runspec命令提供的输入指令。基准被分成完全相同的三组:

Deduplication-sensitive基准：相同的模块平均超过50%以上；

Deduplication-friendly基准：相同的模块平均在20%到50%之间；

Deduplication-insensitive基准：相同的模块平均低于20%。

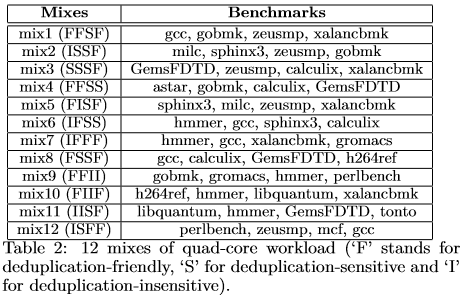
下图表格展示了各个分组以及每个基准系统中重复的百分比、每一千条指令的LLC丢失、指令的循环周期、快速到达由基准系统的SimPoint所提供的时间间隙的指令数量。



## 2.8 实验结果

通过分析缓存重复数据删除的性能及开销。

下表展示了四核负载混合工作:

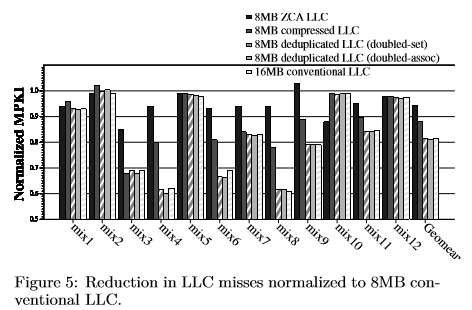


### 2.8.1 性能优化

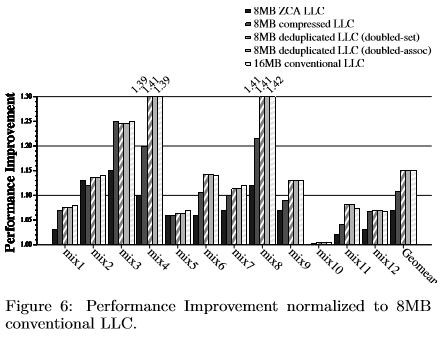
在不重复的高速缓存中，集合的数量与标签阵列的关联性可以同时被增加到放置更多的高速缓存块当中。在压缩存储器中，集合的数量不能增加，并且动态增加关联性的速度是为压缩高速缓存速度的两倍。在一个ZCA高速缓存当中，可以映射高达64MB的空块。

我们将每种技术的性能与一个双尺寸的常规高速缓存进行比较。在我们的实验中，我们展示了一个8MB压缩LLC的性能提升，一个8MB重复数据清除的LLC，一个8MB双关联的重复清除的LLC，一个具有8192条ZCA高速缓存的8MB常规LLC和一个16MB的常规LLC。

下图展示了对于四核负载，每种技术标准化为8MB的LLC后的LLC的高速缓存确实。ZCA高速缓存平均将会减少5.5%的LLC缺失。缓存压缩将LLC未命中减少了12%。在双重LLC当中，缓存重复数据清除将平均未命中率减少了18.5%。在双重关联LLC中，缓存重复数据清除将平均错误率减少了19%。双倍的常规LLC将高速缓存未命中率将少了18.4%。



将减少高速缓存未命中率转换为改进系统性能。下图展示了都是用8MB常规LLC技术性能的提升。ZCA高速缓存将性能提高了6.9%。与标准线相比较，压缩存储的平均加速度为10.8%。在双重LLC中的缓存重复数据清除提高了15%，并且缓存重复数据清除在一个双重相关性的LLC中产生了15.2%的加速。与8MB的标准线相比较，上限为16MB的常规缓存平均加速15.1%。一个12MB的传统LLC速度提高了7.7%，一个14MB的LLC速度提高了8.9%。



总之，重复数据清除的LLC执行效率与双倍大小的常规LLC效率相当。

# 3 Extending Lifetime of SSD in RAID5 Systems through a Reliable Hierarchical Cache

通过一种可靠的分层缓存来延长SSD在RAID5系统中的使用寿命

## 3.1 论文背景与动机

基于NAND闪存的SSD（固态驱动器）自发明以来一直面临使用寿命难题，尤其是在全闪存RAID5系统（本文简称为RAID5系统）中部署以及在随机更新占主导的工作负载下工作时，SSD的磨损度大大增加。

已经提出了许多通过减少对数据，或者对奇偶校验块的更新来延长RAID5系统中的SSD的寿命的解决方案。然而，它们都具有共同的缺点：需要部署额外的大容量永久性存储设备，不仅会极大地增加SSD RAID的成本，还会引发潜在的可靠性问题。

提出与所有现有解决方案不同的思想，通过探索和挖掘RAID5系统内部的潜力来延长SSD的使用寿命。

## 3.2 RAID简介

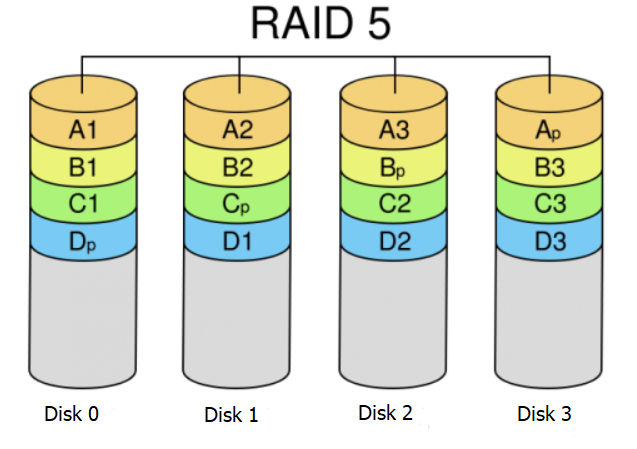
RAID是为了组合小的廉价磁盘来代替大的昂贵磁盘，同时希望磁盘失效时不会使对数据的访问受损失而开发出一定水平的数据保护技术。

RAID就是一种由多块廉价磁盘构成的冗余阵列，在操作系统下是作为一个独立的大型存储设备出现。

RAID可以充分发挥出多块硬盘的优势，可以提升硬盘速度，增大容量，提供容错功能，能够确保数据安全性，易于管理的优点，在任何一块硬盘出现问题的情况下都可以继续工作，不会受到损坏硬盘的影响。

### 3.2.1 RAID5系统介绍

RAID5是一种折中的存储解决方案。一种存储性能、数据安全和存储成本兼顾的存储解决方案。RAID5不对存储的数据进行备份，而是把数据和相对应的奇偶校验信息存储到组成RAID5的各个磁盘上，并且奇偶校验信息和相对应的数据分别存储于不同的磁盘上；当RAID5的一个磁盘数据发生损坏后，利用剩下的数据和相应的奇偶校验信息去恢复损坏的数据。磁盘逻辑结构如下所示：



RAID5数据恢复思路：RAID5阵列中数据的分布与RAID0类似，但是其每个条带上有一个校验块，利用校验块的信息来恢复，恢复有四个重要因素：

1. RAID中每个条带的的大小
2. RAID中硬盘的排列顺序-盘序
3. 校验块的循环方向
4. 数据块的走向

根据RAID的结构属性，把取出来的数据按照顺序衔接成一个镜像文件就是逻辑盘的结构，直接访问重组后的文件就是逻辑盘的数据.

### 3.2.2 SSD优点与缺点

优点：采用闪存作为存储介质，读取速度相对机械硬盘更快；固态硬盘是使用闪存颗粒制作而成，所以SSD固态硬盘内部不存在任何机械部件；固态硬盘上的功耗要低于传统硬盘；固态硬盘没有机械马达和风扇，工作时噪音值为0分贝；大多数固态硬盘可在-10~70摄氏度工作。

缺点：固态硬盘最大容量仅为4TB；固态硬盘闪存具有擦写次数限制的问题，闪存完全擦写一次叫做1次P/E，因此闪存的寿命就以P/E作单位。

### 3.2.3 RAID5中使用SSD的寿命问题

与基于硬盘的传统存储系统相比，基于NAND闪存的存储系统具有更高的性能，更好的抗震性和更低的空闲功耗，因此在企业环境中越来越受欢迎。通过将硬盘替换为固态硬盘，并更新该RAID系统中的一些软件和硬件，从硬盘RAID系统构建全闪存系统是一种方便的方法 。

通常，RAID系统（特别是RAID5系统）中SSD的寿命比无冗余环境中的寿命要短得多。

在RAID5系统中，数据块上的更新可以导致对奇偶块进行更新，即更新会通过RAID5系统中的数据冗余方案来放大，而这些更新会加速SSD的磨损。

**UR \* AR = US**

UR表示RAID层接收的更新块的数量（需要更新块的数量）

AR表示RAID层的更新放大率，即RAID层发出的更新总块与RAID层接收的更新块之间的比率，AR的范围从（1 + 1 /（N-1））到2，其中N是RAID5系统中SSD的数量。

US表示由RAID层发出的更新块的数量。

### 3.2.4 延长SSD寿命的方法

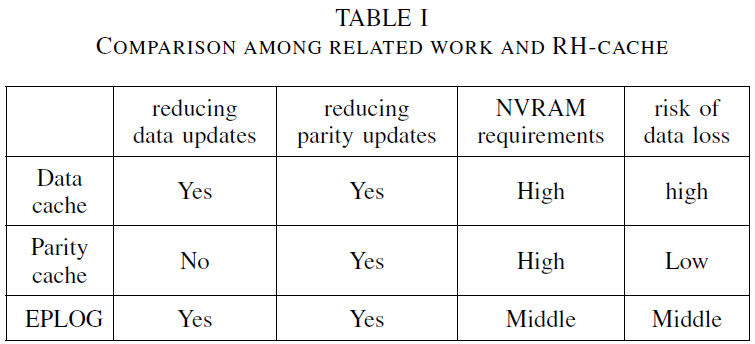
1. 数据缓存
2. 奇偶校验缓存
3. 弹性奇偶校验日志（EPLOG）

数据缓存是减少发送到底层存储设备的写请求的传统方法。通过吸收写入请求，不仅数据块的更新会减少而且更新奇偶校验块也将会减少。但缓存故障可能会导致数据丢失。

尽管奇偶校验缓存比数据缓存更具成本效益和可靠性，但不能减少数据块上的更新，从而失去进一步提高SSD寿命的机会。

EPLOG的技术也可以减少奇偶校验块上的更新。通过保持旧版本和新版本的数据块，旧的奇偶校验块不需要立即更新。同时，新版本的数据块可以合并为大量的顺序写入。但其需要查找驻留在NVRAM中的注册表，如果注册表发生故障，则会导致数据的丢失。

三种方法比较对比如下表所示：



结果表明，现有的解决方案都不能同时满足效率，成本效益和可靠性的要求，因此，RH-cache有动力提出。

## 3.3 RH-Cache设计目标

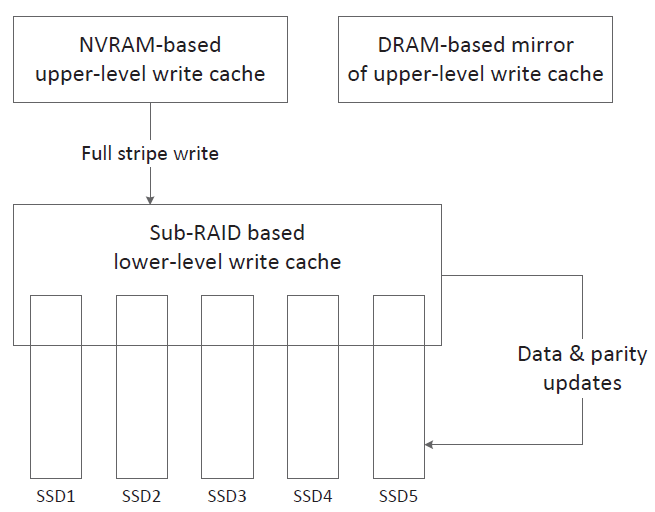
减少数据和奇偶块的更新。为了缓解RAID层引起的SSD寿命问题，我们的目标是在RH-cache中减少UR和AR。

保持RAID5系统不受任何设备故障的影响。RAID5系统旨在提供可靠的存储服务，即对RAID5系统的任何改进都不应引起其他可靠性缺陷。为了保持RAID5系统的可靠性，RH-cache将被设计为容许任何设备故障。

减少NVRAM。延长SSD寿命的关键动机是降低存储系统的TCO（Total Cost of Ownership，总体拥有成本）。较大的NVRAM意味着更高的TCO。NVRAM中误码引起的数据丢失的概率与NVRAM的大小成正比。

## 3.4 RH-cache的总体架构

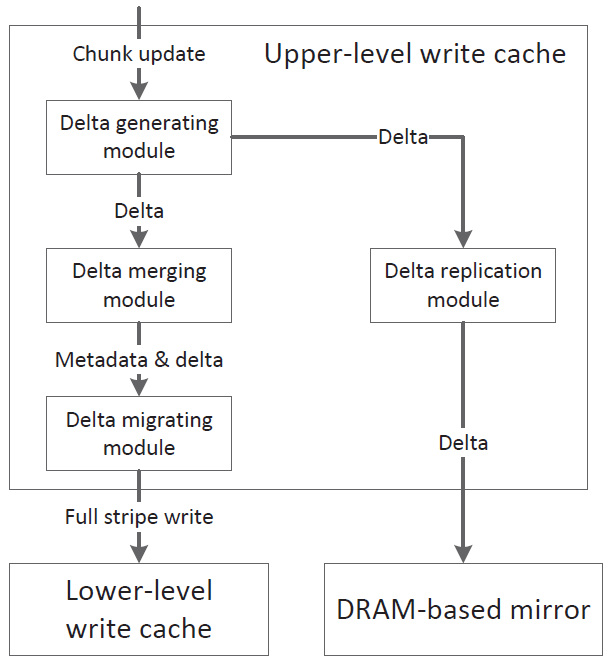
如下图所示：上层写缓存保持旧数据块和新数据块之间的压缩增量（compressed deltas），将这些增量合并为完整的条带，并将完整的条带写回到较低级别的写入缓存。以这种方式，随机块更新可以合并为全条带写入，因此可以减少AR。较低级别的写入缓存保留压缩的变化量，而不是数据块的新版本。由于写入工作负载中的内容本地化，这些压缩增量（compressed deltas）的数量通常远小于数据块的新版本的数量。因此，可以借助较低级别的写入缓存来减少UR。



RH-cache中基于NVRAM的高级写高速缓存和基于RAID子系统的低级写高速缓存

### 3.4.1 高级写缓存

如下图所示：



上层写缓存软件模块图

增量生成模块

增量生成模块将从底层SSD RAID中提取旧版本的请求块。然后，如果请求是全部块更新（请求大小等于块大小，并且请求是块对齐的），则整个请求是所请求的块的新版本。否则，应根据以下程序准备所请求块的新版本：i）在高级和低级写高速缓存中查找所请求块的最新压缩增量；ii）解压缩增量并使用增量和旧版本来生成所请求的块的最新版本；iii）使用该请求来更新所请求的块的最新版本以生成所请求的块的新版本。增量可以通过对所请求的块的旧版本和新版本进行异或来生成。最后，增量被压缩并发送到数据合并模块。

增量迁移模块

当在NVRAM中累积了N-1个数据块（包括增量和它们的元数据）时（𝑁是条带中的数据块的数量），增量迁移模块从这些数据块中生成奇偶块，并将所有数据和奇偶校验块写进低级写缓存的完整条带。然后，可以循环存储这些数据块的高速缓存空间来存储新的增量和元数据。

增量复制模块

在高级高速缓存中生成增量之后，它将被复制到基于易失性存储器（例如，RAID5系统中的DRAM）的另一个镜像高速缓存中，以避免高级写入高速缓存中的单点故障。

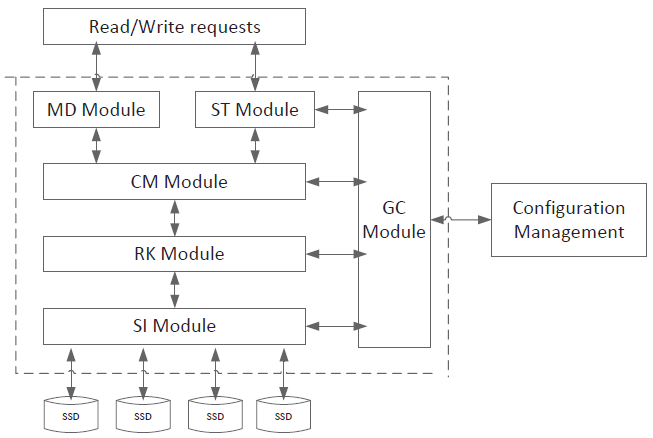
## 3.4.2 低级写入缓存模块

在DRAM中维护一个映射表，以记录上层和下层写高速缓存中的最新压缩增量。当需要为增量生成模块准备块的先前版本时，查找映射表查找块的最新压缩增量。当数据块被更新时，将该数据块的先前压缩增量标记为DRAM中的无效，并更新映射表

增量提交模块将存储在子RAID中的增量提交给底层SSD RAID。基本的提交单元是在子RAID中的一个条带，为了减少提交引发的更新，我们可以使用最少有效的增量（贪婪）来提交一个条带。

当位于DRAM中的映射表被损坏时，增量修复模块扫描NVRAM和子RAID来用上述元数据重建映射表。

### 3.4.3软件RAID控制器体系结构



全局配置（GC）模块

其主要功能是提供网页界面进行交互，监控RAID的运行情况。用户可以通过GC模块创建和管理各种级别的RAID阵列。

SCSI目标（ST）模块。它通过ISCSI协议将用户的控制请求（如读/写请求）发送到目标存储系统。

多盘（MD）模块。它是一个本地块设备驱动程序模块。MD模块捕获本地请求并将分析发送到CM模块。

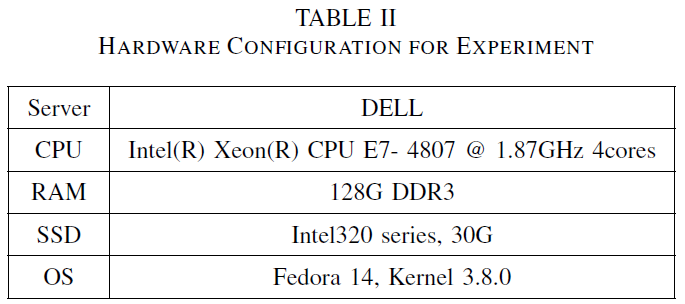
高速缓存管理（CM）模块。它是软件RAID控制器中的缓存管理模块。CM模块的主要任务是高效地缓存数据和管理数据。当请求的数据在缓存中时，数据和响应立即返回到上层。如果请求的数据不在高速缓存中，则向RK模块发送请求。

RAID内核（RK）模块。该模块包含RAID算法各个层次的主要功能，如RAID0，RAID1，RAID10，RAID5，JBOD。数组创建，数组初始化和数组重构都在RK模块中。来自CM模块的请求被映射到相应的RAID算法，然后发送到SI模块。

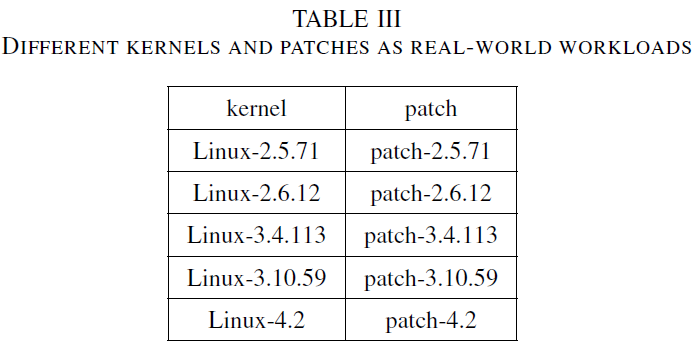
SCSI启动器（SI）模块。它处理特定的存储设备。SCSI接口用于扫描磁盘并访问存储的数据。

## 3.5 硬件要求

### 硬件配置表



RH-cache中的小型NVRAM在现阶段由RAM来实现。SSD RAID由5个Intel 320系列SSD组成。用户空间为30GB，子RAID空间为8GB。条纹深度设置为16KB。3.5.2 基于不同内核的实验。



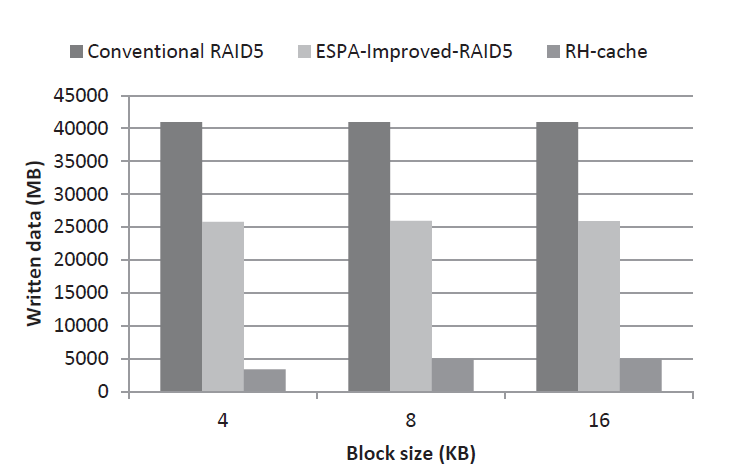
基于不同的内核跟相关配置表

实验中采用I / O基准和真实世界的工作负载。

I/O基准。FIO用于I / O基准测试。这是一个I / O工具，旨在用于基准和压力/硬件验证。

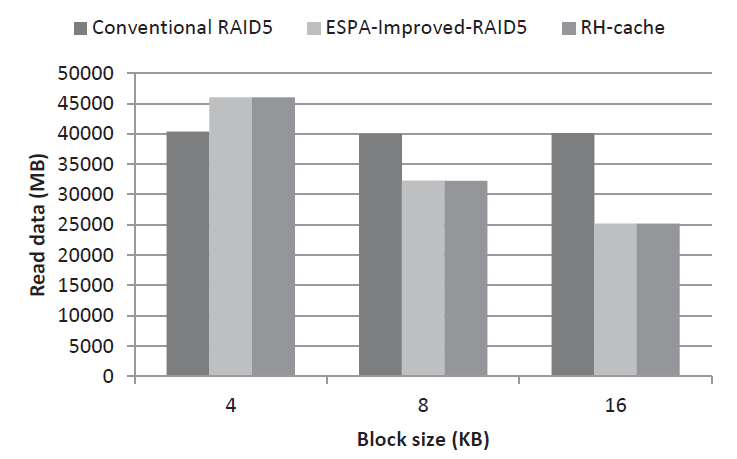
真实世界的工作负载。应用修补程序会产生大量的随机写入。因此，在现实世界中对RH-cache进行评估。

## 3.6 实验评估



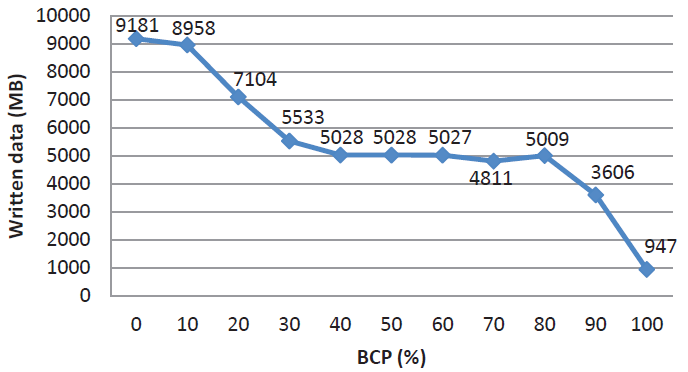
传统RAID，ESAP改进RAID5和RH缓存的总写入

RH-cache的块大小，总写数是最少的，几乎是ESAP-Improved-RAID5的1/5和传统RAID5的1/8。请注意，当有5GB的数据请求时，RH-cache的写入量小于5GB，而传统的RAID5则高达40GB。这是因为RH缓存吸收数据更新和延迟奇偶校验。除此之外，数据在被保存在RH缓存之前被压缩。在这种情况下，一个数据块可以存储几个更新的数据块，而只有一个数据块在非压缩方案中。因此写入SSD的数据实际上少于上层写请求。



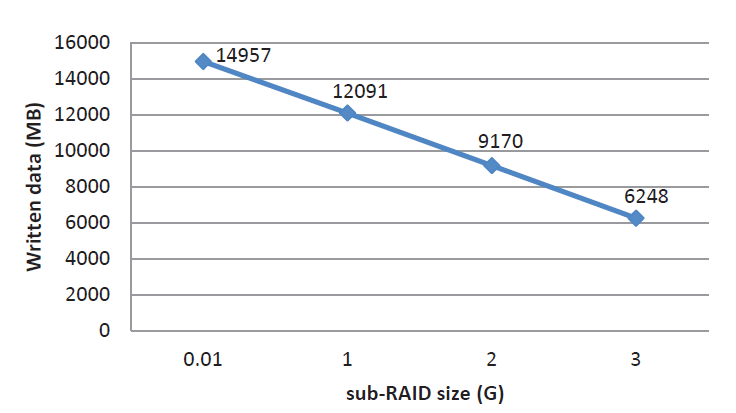
基于不同块大小的对比

当块大小变大时，基于ESAP改进的RAID5读取和RH-cache读取小于传统的RAID5。根据本地原则，同样的数据可以被访问多次。与传统的RAID5不同，它可以延迟两次读取和两次写入，并且可以吸收对相同逻辑数据的更新。结果，数据和奇偶校验的更新都减少了。



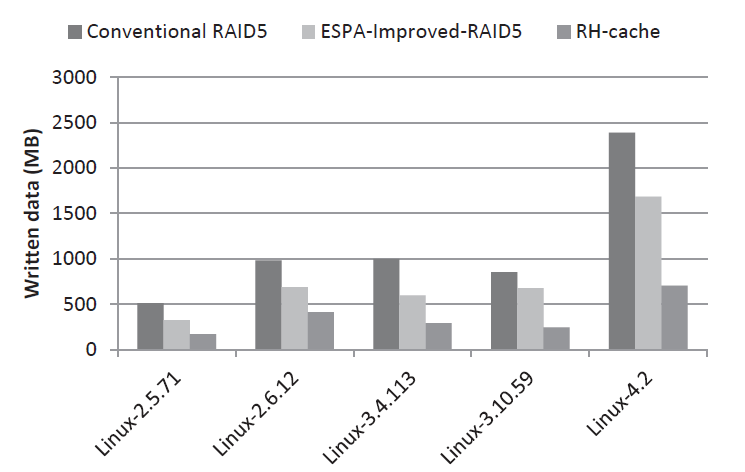
相似性对数据压缩的影响

随着相似度变高，写入的数据减少。当BCP在35％到80％之间时，实际写入的数据几乎等于上层请求5GB。如果BCP低于35％，则由于压缩大小大于非压缩大小，因此需要将更多数据写入SSD。即使当BCP为0时，在旧数据和新数据之间没有相似性，压缩方案不起作用的情况下，总写入仍然小于40GB。这是因为RH-cache仍然可以吸收数据更新并延迟奇偶校验。



不同子RAID大小对总写入的影响

随着子RAID大小的增加，总写入线性减少。当子RAID具有更大的尺寸时，它吸收更多的数据更新并延迟更多的奇偶校验。即使在RH-cache中只有一个小的NVRAM，RH-cache仍然比传统的RAID5和基于ESAP改进的RAID5更好。



基于ESAP改进的RAID5的总写入是传统RAID5的5/8。

在RH-cache中写入的总数几乎是传统RAID5的30％，基于ESAP改进的RAID5的50％。

## 3.7 总结

海量存储课程的学习，使我对海量存储技术有了一个全新的认识，加深了基于存储的概念性总结，了解了不同的存储类型及其构成方式，比如块存储包含单机块存储跟分布式块存储，文件存储，对象存储等。通过阅读相关论文，拓展了自己横向的知识面，对相关技术知识也有了更加深入的认识。

本次提交的论文总结报告，由两篇论文组成，一篇来自于我们小组演讲PPT时的文章，另外一篇来自于国际会议选文。两篇文章都涉及存储的相关知识，一篇讲述的是重复数据删除来保证最高效使用内存的技术，一篇讲述的是基于分层缓存来延长SSD使用寿命的技术。

同时，在论文阅读与选题方面，老师也给予了我们一定的指导，在不懂得方面，老师也及时的给以解答，最后，感谢老师的悉心指导。