**SmartDedup: Optimizing**

**Deduplication for Resource-constrained Devices**

**（SmartDedup：为资源受限设备优化重复数据删除）**

姓名：童帅 学号：M201973383 班级：计算机技术硕1908 论文号：8

# 背景介绍

这篇文章发表于2019年的USENIX Annual Technical Conference会议上，作者是来自于亚利桑那州立大学的杨奇瑞、金润玉和赵明。他们隶属于亚利桑那州立大学（ASU）的虚拟化基础架构，系统和应用程序研究实验室（VISA）（Research Laboratory for Virtualized Infrastructures, Systems, and Applications）。虚拟化是一种启用技术，可用于创建重要的新系统抽象并解决当今计算系统面临的各种挑战。自主性是通过自动管理和优化系统来处理计算系统中日益增加的复杂性的关键。VISA研究实验室的基本目标是探索虚拟化和自主性方面的创新技术，以便有效地利用大规模，动态和复杂的计算系统中的资源，并支持对挑战性应用程序进行高效，强大和安全的计算不同的域。目前，该团队专注的项目通常与云计算，高性能计算和大数据以及操作系统和存储系统有关。

该团队目前主要的项目有：

1. GEARS：异构和动态数据高效节能大数据研究的基础架构

该项目正在开发所需的计算基础架构，以支持GEARS（一种高能效的big-datA研究系统）用于使用异构计算和存储资源来研究异构和动态数据。旨在将GEARS成为基于凝聚力共同设计的软件和硬件组件的一种高能效的大数据研究基础架构。

1. DataStorm：端到端灾难规划和相应的数据启用系统

数据驱动的模型和计算机模拟可用于灾难的防备和响应，在预测灾难的演变以及通过各种干预措施有效地管理紧急情况方面，可以发挥关键作用。该项目将通过多方面的研究来增强灾难响应和社区的适应能力，以创建一个大数据系统，以必要的数量，速度和多样性支持数据驱动的模拟，并整合和优化灾难管理的关键方面和决策。

1. DM-CACHE：云计算系统的动态存储缓存

块级分布式存储系统（例如SAN，iSCSI）通常用于新兴的云计算系统中，以提供虚拟机（VM）存储。它们允许在不同主机之间进行快速VM迁移，并利用此类存储系统中可用的典型容错措施（例如RAID）来提高VM可用性。但是，随着云系统的规模和托管VM的数量迅速增长，共享块级存储系统的可伸缩性成为一个严重的问题。该项目建议通过使用客户端存储来实现块级缓存并利用VM数据访问中可用的数据局部性来解决此问题。通过利用VM主机上可用的SSD等快速存储设备的容量，这种方法可能显着提高VM的性能以及共享存储系统上的负载。

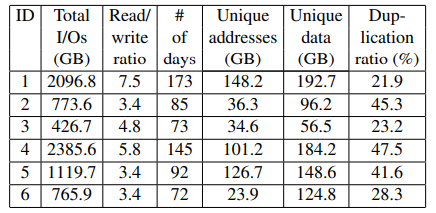
# 论文综述

### 1.问题介绍

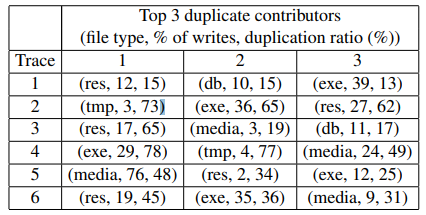
该论文提出的原因在于：第一，随着智能设备的普及和物联网的发展，数据的总量越来越大、数据的产生速度越来越快；第二，智能设备一般可利用的资源有限。两者的矛盾引发出了新的问题——这对于资源受限设备上的闪存技术来带了极大的挑战。然后作者提出了一种可能的解决方案——重复数据删除。重复数据删除被广泛利用于云技术，但是很少被利用于资源受限设备上。利用重复数据删除，首先可以消除I/O的冗余以提升性能，其次可以减少闪存的写入以增强耐用性，最后可以删除冗余数据以提高利用率。

### 2.可行性分析

虽然，重复数据删除带来了这么多利益，但是，在没有进行数据分析前，都不能确定重复数据删除是否真的可行。比如：数据量很大而且数据产生速度快，但是数据重复率很低，此时进行重复数据删除的作用就无法体现，甚至可能降低系统性能。再比如：数据的重复性具有一致性，可能只有特定的数据重复率极高，而其余数据重复率较低。针对这两个问题，作者进行了样本数据采集，然后对样本数据进行了分析。具体数据如下：



对于此表，可以看出重复数据比率确实很高，均高于21.9%，所以重复数据确属存在且比例较高。接下来又针对数据的类型进行了分析，结果如下表：



对于此表，可以看出重复数据是普遍存在的，重复数据的类型是不确定的，只针对特定的文件类型进行重复数据删除是不能解决问题的。

### 3.方法设计

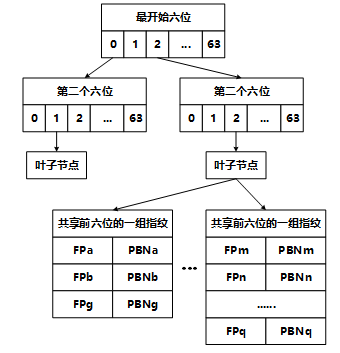
针对这个问题，文中提出了三个设计原则来指导系统的设计：

1. 由于设备的限制，重复数据删除应该被尽可能多的执行
2. 内存的容量有限，所以数据结构应该尽可能少，同时也要使用磁盘空间来进行弥补
3. 设备通常能源有限，重复数据删除需要对能源和资源进行自适应调整

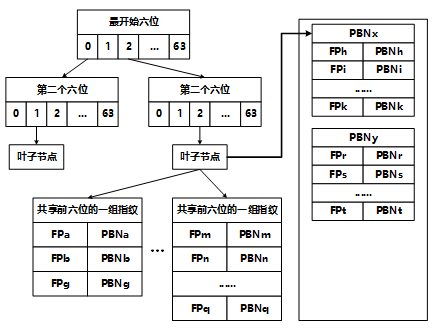
针对以上三个指导原则，作者进行了如下设计：

#### 内存指纹存储和磁盘指纹存储

其中，内存指纹存储按照如下的一个前缀树进行存储：



为了节省空间，作者采用的方法是索引到一组指纹，而不是单独的一个指纹。其中，每组指纹共享相同的前缀，并且每组指纹内部通过链表进行连接。作者探讨了可能发生的问题，链表可能长度过长，导致搜索指纹时间较长，根据分析可得，有如下两个原因使得链表长度不会过长。第一，内存中只保存重要的指纹，通常占用的空间很少，超出一定空间后会进行指纹迁移。第二，用于指纹识别的哈希函数趋向于将指纹均匀分布在不同的组。这两个原因导致每组链表不可能太长。

内存中只保存了重要的指纹，那么“不重要”的指纹就需要保存在磁盘指纹存储中，磁盘指纹存储如下图所示：  


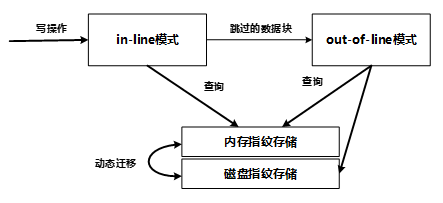
磁盘指纹的索引方式就是在内存指纹前缀树的相应叶子节点中添加磁盘文件的物理块号，这样就保证内存指纹索引和磁盘指纹索引数据结构的一致性。这样既可以节省空间，同时也有利于指纹的迁移。

指纹迁移的目的是动态更新指纹的存储位置，将重要的指纹存储在内存中，将不重要的指纹存储磁盘中。既然提到了重要性，首先就应该对重要性做出定义，即何为重要的指纹，文中采用的是LRU算法，即最近使用的指纹为重要的指纹。

当移出指纹时，为了节省I/O开销（一次移出操作包括两次I/O，将磁盘指纹数据读取到内存中，将更新的文件协会到磁盘中），作者采用一组指纹移动的方法进行，即以组为单位进行指纹的移出。当然，为了保证内存的空间，移入指纹时任然一个一个依次进行。

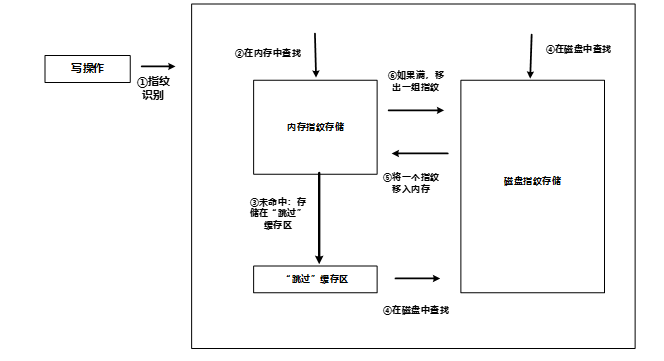
#### In-line模式和out-of-line模式

文章使用的是in-line模式和out-of-line模式混合进行的重复数据删除，具体工作方式如图所示：

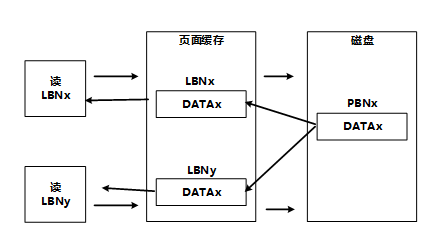


作者提出的重复数据删除的方法是，将LBN到PBN的一对一映射改为多对一映射。in-line模式在I/O路径中工作，可以立即消除重复的写入，但是受限于设备上的可用资源。Out-of-line模式在后台工作，可以缓慢彻底地消除重复数据，但是对性能和持续工作能力方面的改进有限。

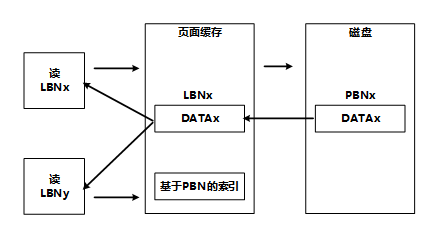
对写数据过程，采用下图的步骤进行重复数据的删除：



对于读数据过程中的重复冗余，作者也采取的一定的办法：  
正常情况下的读数据过程如图所示：



由于页面缓存是通过LBN索引的，所以无法避免重复读取相同的数据，因为无法知道数据是否已经被导入进内存中。作者提出了下图方法来消除这个冗余：



作者在内存中创建了基于PBN的索引，是从PBN到页面缓存中相应页面的索引。在重复数据读取时，基于这个索引表，就能找到页面缓存中相应的数据，从而避免重复读取。

#### 针对设备当前能源的自适应调整

针对自适应调整，作者从如下两个方面进行了考虑：

基于资源的可用性：保持cpu和磁盘不会过载，可以与电池寿命成比例，在低电量的情况下完全禁止。

基于当前的重复率级别：重复率级别下降时，缓慢降低重复数据删除效率。重复率级别上升时，迅速上升重复数据删除效率。

基于这两个方面，作者给出了如下实现：

对于out-of-line模式：每隔一段时间调整处理的块数

对于in-line模式：针对总的N个请求，动态调整，选取其中的n（n<=N）个请求进行处理。

### 4.总结

文章针对数据量大、数据增长速度快和设备资源受限这个矛盾问题，提出了重复数据删除的解决方案。作者使用内存指纹存储和磁盘指纹存储来存储指纹，并且公用同一个索引，在保持高效的同时只占用少量的空间。其次，作者为了尽可能多的进行重复数据删除以提高删除率，采用了in-line模式和out-of-line模式共存的方法。In-line模式高效地进行重复数据删除，out-of-line模式低效但彻底，二者相辅相成。最后，针对设备的不同使用情况，重复数据删除会自适应调整，在满足不超过系统负载的情况下，尽可能的进行重复数据删除。