计算机科学
COMPUTER SCIENCE

基于重复数据删除的分层存储优化技术研究进展

姚子路, 付印金, 肖侬

引用本文

姚子路, 付印金, 肖侬. 基于重复数据删除的分层存储优化技术研究进展[J]. 计算机科学, 2025, 52(1): 120-130.

YAO Zilu, FU Yinjin, XIAO Nong. Research Progress on Optimization Techniques of Tiered Storage Based on Deduplication [J]. Computer Science, 2025, 52(1): 120-130.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231200011

基于重复数据删除的分层存储优化技术研究进展

姚子路[1]　付印金[2]　肖　侬[1,2]
1 国防科技大学计算机学院　长沙 410073
2 中山大学国家超级计算广州中心　广州 510006
(569659537@qq.com)

摘　要　随着全球数据量的爆炸式增长以及数据多样性的日益丰富,单一介质层的存储系统逐渐不能满足用户多样化的应用需求。分层存储技术可依据数据的重要性、访问频率、安全性需求等特征将数据分类存放到具有不同访问延迟、存储容量、容错能力的存储层中,已经在各个领域得到广泛应用。重复数据删除是一种面向大数据的缩减技术,可高效去除存储系统中的重复数据,最大化存储空间利用率。不同于单存储层场景,将重复数据删除技术运用于分层存储中,不仅能减少跨层数据冗余,进一步节省存储空间、降低存储成本,还能更好地提升数据I/O性能和存储设备的耐久性。在简要分析基于重复数据删除的分层存储技术的原理、流程和分类之后,从存储位置选择、重复内容识别和数据迁移操作3个关键步骤入手,深入总结了诸多优化方法的研究进展,并针对基于重复数据删除的分层存储技术潜在的技术挑战进行了深入探讨。最后展望了基于重复数据删除的分层存储技术的未来发展趋势。

关键词:重复数据删除;分层存储;存储位置选择;重复内容识别;数据迁移
中图分类号　TP311

Research Progress on Optimization Techniques of Tiered Storage Based on Deduplication

YAO Zilu[1],FU Yinjin[2] and XIAO Nong[1,2]
1 College of Computer Science and Technology,National University of Defense Technology,Changsha 410073,China
2 National Supercomputer Center in Guangzhou,Sun Yat-Sen University,Guangzhou 510006,China

Abstract　With the explosive growth of global data volume and the increasing diversity of data,storage systems with a single media layer are gradually unable to meet the diverse application demand of users. Tiered storage can classify and store data into storage layers with different access latency,storage capacity,and fault tolerance based on the importance,access frequency,security requirements,and other characteristics of the data. It has been widely applied in various fields. Deduplication is a big data reduction technique that can efficiently remove duplicate data from storage systems and maximize storage space utilization. Unlike single storage layer scenarios,applying deduplication to tiered storage can not only reduce cross-layer data redundancy,further save storage space and reduce storage costs,but also improve data I/O performance and storage device durability. After a brief analysis of the principle,process,and classification of deduplication based tiered storage,this paper starts with three key steps:storage location selection,duplicate content identification,and data migration operation. It summarizes the research progress of many optimization methods and explores the potential technical challenges of deduplication based tiered storage. Finally,the future development trends of deduplication based tiered storage is prospected.

Keywords　Deduplication,Tiered storage,Storage location selection,Duplicate content identification,Data migration

1　引言

近年来,存储领域发展面临的压力与日俱增。一方面,大数据和人工智能技术的发展直接导致了用户数据量的爆炸性增长,据IDC发布的统计和预测[1]:全球数据量规模将从2022的103.66ZB增长至2027年的284.3ZB。传统单一设备层的存储系统,例如单个的客户机、磁盘、服务器逐渐无法负荷庞大的数据量,而磁盘阵列、服务器集群和云存储等大容量存储设备成为大规模数据的主要流向。另一方面,计算机技术与不同领域的结合使得数据的复杂性也呈几何级提升,不同的数据有不同的特征,包括访问频率、重要程度、安全性需求等,它们被统一存放在一个存储层中,即使容量上能

到稿日期:2023-12-01　返修日期:2024-04-27
基金项目:国家重点研发计划(2022YFB4500304);国家自然科学基金(62332021,61832020)
This work was supported by the National Key Research and Development Program of China(2022YFB4500304) and National Natural Science Foundation of China(62332021,61832020).
通信作者:肖侬(xiaon6@mail.sysu.edu.cn)

满足，也不利于管理。因此，针对不同数据特性安排不同类型的存储介质专门存储成为主流的处理方法。在这种环境下，分层存储技术崭露头角，其不仅解决了单一存储层容量告急的问题，还解决了多样化数据的管理问题，成为了广泛运用于各个领域的通用解决方案。

分层存储虽然方便，但考虑到成本、性能和存储介质的寿命等问题，仍有极大的优化空间。在这种情况下，将重复数据删除技术运用于分层存储中，排除跨层存储管理中冗余的数据，保证内容的唯一性，不仅能进一步节省存储空间、降低存储成本，也能减缓存储介质寿命的损耗速度，是非常实用的优化方法。本文总结和提炼了基于重复数据删除的分层存储优化技术的代表性工作，旨在为研究者们准确理解和把握未来创新方向提供借鉴和参考。

本文第 2 章简要介绍基于重复数据删除的分层存储相关知识，第 3－5 章从 3 个关键步骤入手介绍基于重复数据删除的分层存储技术相关优化方法的研究进展，第 6 章对基于重复数据删除的分层存储优化技术未来的发展前景做出展望。

2 基于重复数据删除的分层存储简介

2.1 背景知识

2.1.1 数据生命周期管理与存储系统

数据生命周期管理是一种在从数据输入到数据销毁的整个生命周期内管理数据的方法，被广泛运用于各行业中。数据根据不同的条件分处不同的阶段，随着其完成不同的任务或满足特定要求而逐次经历这些阶段。IBM[2] 将数据的生命周期分为数据创建、数据存储、数据共享与使用、数据归档和数据删除 5 个阶段。对于存储系统来说，其关注的重点是数据存储介质选择、存储架构设计、存储位置调整、存储安全保证，尤其是存储系统的性能直接影响着数据生命周期管理的决策。

2.1.2 分层存储

分层存储，也称为层级存储管理，广义上讲，就是将数据存储在不同层级的介质中，并在不同的介质之间进行自动或者手动的数据迁移、复制等操作。同时，分层存储也是数据生命周期管理的一个具体应用和实现。

分层存储的概念在计算机发展的早期就可窥见一二，早期的存储层次结构就是标准的分层存储的一个基础版本。随着技术的发展和数据量的骤增，传统的本地存储层次结构的划分无法满足用户的需求，云存储逐渐进入分层存储的考虑范围，成为了更低一级的存储层，如图 1 所示。

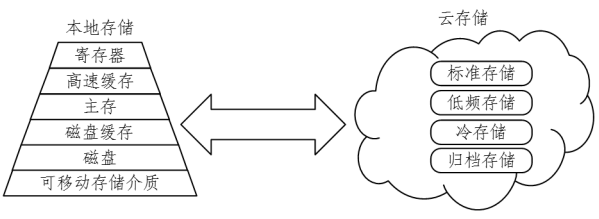


图 1 一种分层存储结构

Fig. 1 Tiered storage architecture

云存储根据数据存储时间和访问频率可进一步分为云端的标准存储、低频存储、冷存储和归档存储等，为用户提供不同质量、不同价格的服务。理论上的分层过程通过数据的一系列特征确定数据适合存放的位置，常用的缓存-内存模型就可被视为一个以数据的访问频率为主要分层特征的简单的两层存储系统，而在更复杂的分层存储中需要考虑更多数据本身的特性和用户访问信息以进行更科学的分层。

2.1.3 重复数据删除技术

重复数据删除是一种高效的数据缩减技术，被广泛运用于各种空间敏感型的应用中，可将存储系统中的冗余数据按一定粒度进行去重，独特数据仅占唯一存储空间，其余包含相同数据的地址都通过指针指向唯一数据所在位置。该技术在保证数据信息不混淆的前提下最大程度节省了存储空间[3]。

目前主流的重复数据删除的分类方法一般从 4 个方面入手[3]：1）从重复数据删除的操作粒度区分；2）从重复数据删除的操作时机区分；3）从重复数据删除的精确度区分；4）从重复数据删除的删除域区分。

根据重复数据删除的操作粒度，可分为文件级重复数据删除、块级重复数据删除和字节级重复数据删除。文件级重复数据删除将两个文件整体视为重复内容的判断单位；块级重复数据删除将文件分为不同的数据块，每次查询单独数据块内容是否重复；字节级重复数据删除，顾名思义，就是每次判断单个字节内容是否重复并进行相应处理。文件级重复数据删除操作简单而迅速，但粒度太大导致数据缩减率低，重复数据删除效果不好；字节级重复数据删除能成功避免大量冗余数据的存储，但实现难度高，开销大[4]。因此，块级重复数据删除基于其平衡性成为研究的主流。

根据重复数据删除时机的不同，可分为 inline 和 offline 两种。inline 指在数据实际存入存储设备时，或是在数据从一个层迁移到另一个层的过程中进行重复数据删除，只有独特数据进入目标层；而 offline 是进行完整的数据迁移后再在目标层上运行重复数据删除算法进行数据削减。前者可以节省实际存储设备空间，延长存储设备寿命，但可能受网络吞吐量和计算资源等的限制；后者可合理运用云计算资源等加速重复数据删除过程，但更依赖存储资源，对空间和设备寿命的影响较大[4]。

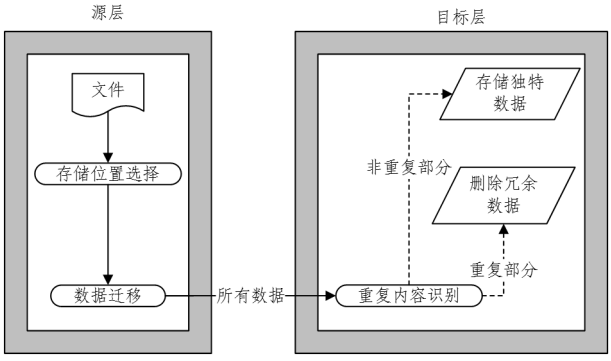
根据重复数据删除的精确度，可分为精确重复数据删除和非精确重复数据删除。精确重复数据删除执行严格的重复内容检测过程，旨在消除所有被判定为重复内容的数据块，保证每个数据仅被储存一次；非精确重复数据删除利用文件相似性提高重复数据删除效率，通过设计相似性识别算法定位重复内容，所有被算法识别为相似的块仅保留一份，这样不需要精确的哈希值计算、索引和比对，能极大提升重复数据删除的速度，但可能造成误删的问题，不适用于准确性和安全性要求较高的情况。

为了应对分布式环境，重复数据删除可按照删除域分为全局重复数据删除和本地重复数据删除。本地指仅在一个节点内部进行重复数据删除，优点是避免了节点间通信的开销，缺点是牺牲了一部分重删率。全局重复数据删除通过搭建一个全局数据库存放所有独特数据的指纹信息，保证所有节点间不存在冗余数据，更符合分层存储系统的要求。

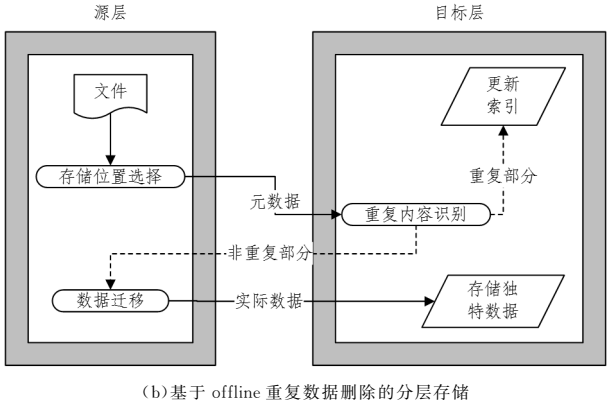
2.2 基于重复数据删除的分层存储原理

与简单的将多个包含重复数据删除功能的存储层相加不同，基于重复数据删除的分层存储系统应该具备保证整个系统不同存储层总共只保留一份独特数据的能力，前者虽然在每个层上实施重复数据删除操作，但仍会存在大量冗余数据在每个层都有一份副本的情况，造成资源浪费。例如一个被判断为几乎不再会被访问的文件，将其移动到云端的归档存储中即可，在其余层存在的冗余备份可以在不影响安全性的前提下直接删除。所以说，基于重复数据删除的分层存储优化技术是在分布式系统中进一步节省存储空间的必要选择。

基于重复数据删除的分层存储根据一定的分类标准，将数据分别存储在合适的存储层中，并根据策略的变化在实际数据迁移前后运行重复数据删除算法，最终实现节约存储空间的目的。为了优化基于重复数据删除的分层存储的效果，需应对 3 个方面的技术挑战：1）如何确定数据存放位置？2）如何高效确定冗余数据？3）如何实施迁移操作？针对这 3 个问题，可将基于重复数据删除的分层存储大致分解为存储位置选择、重复内容识别和数据迁移三大关键步骤，如图 2 所示。



(a)基于 inline 重复数据删除的分层存储



(b)基于 offline 重复数据删除的分层存储

图 2 三大关键步骤流程

Fig. 2 Process of three key steps

此处考虑对存储系统中已有的文件进行基于重复数据删除的分层存储。对于新进入的数据，由于缺乏历史信息等，分类不好进行，可以先暂存快层观察一段时间后再按上述流程执行。其中，存储位置选择模块根据用户需求和数据特征，综合考虑时间、空间、成本等为传入数据智能分层。对于 inline 重复数据删除，重复内容识别模块将文件数据特征元数据（常用指纹）从源存储层发送到目标层，通过与目标层实际存储内容的元数据对比，对于重复部分，仅添加索引到对应地址并修改相应元数据，而不再次存储冗余数据；对于非重复部分，返回该部分信息给源存储层，源存储层由此区分目标层需要的独特数据及目标层已有的冗余数据，并将独特数据内容经由数据迁移模块移至目标层；对于 offline 重复数据删除，源存储层将所有数据经由数据迁移模块移至目标层，并在目标层上进行重复内容识别，保留独特数据，对冗余数据进行回收操作并增加指向已有数据的索引。为了便于章节的组织，本文后续对优化方法的讨论顺序以 inline 为例。

基于重复数据删除的分层存储系统仍然要面对性能与成本的权衡问题，对一个完整的分层存储系统统一使用一种重复数据删除方法难以应对复杂的用户需求变化和工作环境变化。目前主流的解决方案是针对不同的存储层设计不同级别的重复数据删除策略，在性能敏感的存储层采用文件级重删、非精确重删等加速对比速率，并在文件向外迁移时采用 offline 重删将关键比对步骤转移到目标层进行；对于不太强求性能而比较注重数据削减率的层，可以使用全局 inline 块级精确重删，利用大量计算资源集中换取最高的空间节省比例。

可以说，重复数据删除技术为分层存储系统解决了关键的性能和成本问题，分层存储系统为重复数据删除技术的发展提供了新的平台和应用场景，如何优化基于重复数据删除的分层存储也引起了广大学者的关注。

3 针对存储位置选择的优化技术

数据在存储系统中存在一段时间后，会产生许多特征元数据，如时间戳、大小、类型、所处路径、访问情况等信息。借助这些信息，设计不同的存储位置选择算法，可以大幅优化分层存储的效率。该步骤研究成果主要集中在从分层存储刚兴起时就遥遥领先的传统算法上，如何与智能算法对接仍缺少进一步的探索。存储位置选择步骤的主要流程如图 3 所示。

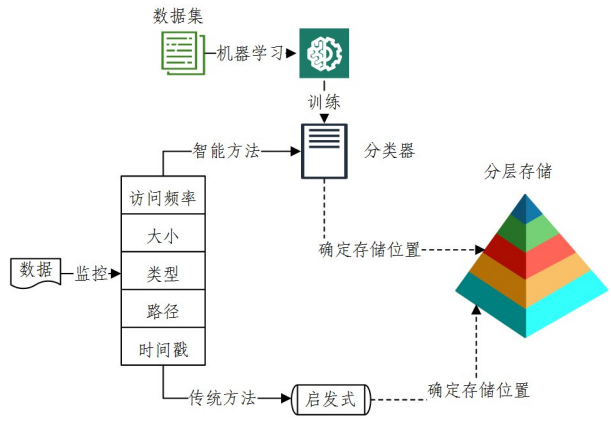


图 3 存储位置选择流程

Fig. 3 Process of storage location selection

3.1 基于文件属性的传统冷热分层

最早的分层算法借助的是数据的访问时间和访问频率等固有特征，频繁访问、近期使用的热数据存放在快层，不太常用的冷数据转移到慢层，也符合局部性要求，但也只能在简单的环境下提供有限的优化效果。

Lee 等[5]在研究中表明，现有分层内存系统基于静态

阈值的冷热划分稍显僵硬，理想的分层系统应考虑快层容量和真实的热数据大小，基于此目的，他们开发了一个动态进行冷热页面分层存储和页面大小决策的系统 MEMTIS。MEMTIS 选择页面存储位置的底层逻辑仍然是访问频率，但其额外设计了页面访问直方图用以统计每个周期内不同访问次数的页面分布情况，页面分布直方图将所有页面大小分为 16 个指数级递增的区间，每个周期统计每个区间中的页面数量，由大到小依次装入快层直到区间 x 无法完全装入，此时将 hot page 的阈值 T_{hot} 定为 x，根据剩余快层容量将 warm page 阈值 T_{warm} 定为 T_{hot} 或 $T_{\text{hot}}-1$，cold page 阈值 T_{cold} 定为 $T_{\text{warm}}-1$，每个周期根据真实情况进行冷热页面划分，并为快慢层设计针对冷热页面的升降级迁移算法。经实验证明，MEMTIS 在一定程度上优化了传统冷热分层不够灵活的问题，实现了性能的提升。

成本与性能也能作为分层的标准。对于快速但容量小、价格高昂的 DRAM 来说，PMM 具有容量大且便宜的特点，Hildebrand 等[6]开发了 AutoTM 方法对机器学习中的张量进行分层存储，将 50%～80%的 DRAM 替换为 PMM，用于存储暂时不用的张量，以一个可接受的小幅性能损失换取了巨额的容量提升和成本降低。整个过程被形式化为一个 ILP 问题，且假设具有全局视野，能提前确定张量的迁移方向，以在一定分层条件下最小化系统的执行时间。

随着云存储的普及，在数据存储的过程中不得不考虑云服务的成本问题。对多个大规模云服务厂商 Amazon Web Services、Microsoft Azure、百度云、阿里云、腾讯云的研究表明，对象存储的云定价基于存储环境的冷热存在数量级上的差距[7-11]，如表 1 所列。

表 1 各云服务商对象存储云定价

Table 1 Object storage priceof several cloud service providers

元/GB/月	AWS	Azure	百度云	阿里云	腾讯云
标准存储	0.1810	0.1620	0.1190	0.1200	0.0990
低频存储	0.1340	0.0800	0.0800	0.0800	0.0800
冷存储	0.0300	0.0330	0.0320	0.0330	0.0300
归档存储	0.0134	0.0146	0.0150	0.0150	0.0100

但对于存在冷云上的数据，受限于其吞吐量，获取其所需的时间成本和数据传输成本甚至可能超过存储的成本。为了实现最具成本效益的云分层系统，Kotlarska 等开发了 InftyDedup[12]。InftyDedup 扩展了数据块的元数据信息，使得每个数据块对应的指纹信息中还包含数据块所在容器中的最大文件到期时间以及预估的文件恢复频率。借助这两个信息，设计了如下公式计算该块存入冷云和热云的期望成本，数据的存储位置被确定为成本较低的一方。

$$t=Cost_{\text{insert}}+(Cost_{\text{B/day}}+Cost_{\text{restore}}*FREQ_{\text{restore}})*EXP_{\text{time}} \quad (1)$$

对于第一次存储的块，其获取到的最大文件到期时间和预估的文件恢复频率不准确，Kotlarska 等[12]还设计了基于块引用计数的启发式函数来动态调整二者的值使其与真实值更相符。经过这样设计的重复数据删除框架 InftyDedup 为分层存储系统实现了良好的可扩展性和高成本效益。

Yang 等[13]在云存储层构建了更细化的 RAM-SSD-HD 的分层结构，设计了自适应的重复数据删除算法 EAD 来决定云端数据存放在哪一层中；此外，为了节省昂贵的 RAM 开销，设计了一套可以动态调整 RAM 容量以适应不同的重复数据删除率的标准，通过降采样操作、RAM 扩展操作和热元数据库完善操作，仅将最能保证重复数据删除效果和效率的条目存储在 RAM 中，在与全索引相比性能下降不到 2%的条件下，仅使用了全索引 5%的 RAM 容量和 RAM-HD 架构 25%左右的 I/O 开销。

可以看到，分层算法优化的趋势开始从理想化的 one-fit-all 静态设计转向动态适应真实工作环境，但仍然未能解决评判标准单一化的问题，因此引入分类能力更强、更准确的算法迫在眉睫。

3.2 基于机器学习算法的自动分层

自动分层存储是一种数据存储管理技术，它根据数据的访问模式和重要性，将数据自动分配到不同层次的存储介质中，以实现更高效的数据存储和访问，基本流程为通过机器学习算法预先训练出分类器，直接用分类器进行自动分层。Wang 等[14]在与腾讯的联合研究中发现 QQ 相册中存在大部分仅第一次被存入后就不再被访问的照片，采用机器学习的方法预测仅访问一次的 one-time-access 照片，不将其放入缓存，从根本上减少了缓存写次数，提高了缓存利用率和命中率。对于已知大小的 cache，可根据缓存替换算法的不同计算出平均一张照片被替换出 cache 的平均访问次数，one-time-access 照片实际上代表那些重访问距离大于该次数的照片，将这样的照片存入 cache 将不会发生 cache 命中，故考虑不将其存入 cache。在综合比较随机森林、AdaBoost、决策树、KNN 等 7 种机器学习算法分类的性能[15]后，最终采用性能好且复杂度低的决策树[16]为分类算法。特征选择方面，one-time-access 提取了照片包括所有者相册平均访问次数、所有者活跃好友数、照片访问频率、照片年龄、照片类型、照片大小、设备访问时间、终端类型和近段时间请求共 3 个方面的 9 项数据作为特征进行分类器训练，并设计了历史表用于纠正预测错误的情况。需要特别注意的是：one-time-access 的实现是基于 QQ 相册良好的设计。在实际将机器学习技术运用于文件智能分层时可以发现，大多数文件系统不提供文件的访问频率等关键信息，选择合适的特征或是重新设计数据结构来提高预测的准确性仍是不可忽视的问题，但这种借助机器学习技术的自动分层方法已经为优化基于重复数据删除的分层存储提供了新的思路。

3.3 小结

对近十年存储位置选择方面的优化方法总结如表 2 所列。可以观察到，从 2016 年 one-time-access 提出利用机器学习算法进行照片文件分类之后，并未出现更普适的分层算法，一个很重要的原因在于机器学习算法会引入较高的特征数据获取成本和分类模型训练成本，对于轻量级的系统来说得不偿失。为了满足响应时间尽可能短、服务时间不中断等要求，机器学习算法并未找到取代传统算法的机会。但随着分布式系统的发展、云存储的普及和跨广域网传输的加入，存储位置选择的优化方向将逐渐明确：传统的启发式算法很难兼顾多项数据特征，利用人工智能技术的分类和预测能力会成为

进一步优化分层效率的突破口。

表 2 近十年存储位置选择优化技术

Table 2 Optimization techniques for storage location selection in the past decade

时间	核心思想	优化成果	优化目的
2016	利用决策树的智能分层	One-time-access	提升缓存命中率从而提升性能
2018	自适应调整采样方式	EAD	在保证性能的同时降低存储成本
2020	线性规划方法求解	AutoTM	权衡性能和成本
2023	页面访问直方图协助冷热划分	MEMTIS	提升分层性能
2023	设计新启发式协助分层	InftyDedup	降低存储成本

4 针对重复内容识别的优化技术

确定好数据存放位置后，先会在源层与目标层间建立连接，传递所有待操作数据的元数据信息到目标层，与目标层已存储的数据对比查重，仅将目标层需要的独特内容对应的元数据信息回传，其余建立到已有数据的索引，这一步称为重复内容识别。目前对于重复内容识别的研究呈现百花齐放的态势。传统的计算数据哈希值作为指纹的检测方法仍具有普适性，对于大规模系统指纹预取的优化也是一个永恒的课题；同时，许多学者开始探索代替指纹的特征，将机器学习方法植入相似性对比环节，从而实现更细致的查重。在此之外，对摆脱计算开销和自适应的重复数据删除算法的研究也在稳步前进。重复内容识别步骤的主要流程如图 4 所示。

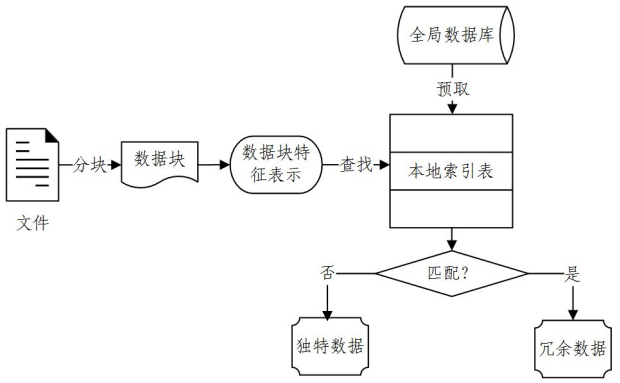


图 4 重复内容识别流程

Fig. 4 Process of deduplicate content identification

4.1 基于指纹的优化

作为重复数据删除最经典的鉴定重复数据的方法，指纹是将目标文件/块/字节存储的内容输入给哈希函数进行计算得到的一个哈希值[17]。由于哈希值具有防碰撞特性，相同的输入会生成相同的指纹，不同的输入大概率会生成不同的指纹，故使用指纹进行重复内容的识别成为大量研究的基础。以指纹为识别基础，需要采取更针对性的方法以适应不同的工作环境，激发了许多优化手段。

4.1.1 非加密哈希指纹

一般情况下，指纹采用的哈希函数应该尽量避免冲突的产生，加密哈希函数的单向性可以保证从哈希值难以推断出原有内容，因此深受学者们青睐。从 MD5，SHA-1，SHA-2 到 SHA-3，都是为降低指纹碰撞概率而逐渐发展出的方法，但哈希函数效果的提升往往伴随着性能的降低，所以选择哈希函数要代入实际的应用场景中。

非加密哈希可以容忍哈希碰撞，计算速度快且占用的空间更少，在特定环境下能起到意想不到的效果。Qiu 等观察 NVM 的 I/O 特性发现，传统的重复数据删除方法并不适用于 NVM，因为快速 NVM 的瓶颈不在 I/O 上，而在 CPU 性能上，复杂的加密哈希算法反而会降低重复数据删除的性能[18]，因此提出了一种针对 NVM 的重复数据删除架构 LightDedup[19]。LightDedup 采用非加密哈希指纹作为块特征，当发现相同指纹时进行更细致的逐字节比对确定重复块，同时加入推测性预取提升性能，并用 region-based 链表存储管理相应的重复数据删除和推测预取的关键元数据。相对于基于加密指纹的 NVM 重复数据删除系统，在重复数据比例较高的情况下，LightDedup 实现了明显的性能提升。

4.1.2 完美哈希指纹

完美哈希[20]指纹指不会产生冲突的指纹，严格来说，其要求事先知道输入数据并定制相应的哈希函数，使得每个输入 1∶1 映射到唯一的哈希值上。为了实现简单，也可通过多个哈希函数组合映射得出不冲突的情况来近似完美哈希函数。Duggal 等[21]扩展了 DELL EMC 的 Data Domain 存储设备的云层部分，且在云层保留了重复数据删除过程。由于 Data Domain[22]本身的成熟性，为了向其中加入云层，做出了以下更改：1）架构方面，无论是本地层还是云层的文件都使用 Merkle Tree[23]来表示，Merkle Tree 的核心思想是将大量数据块分成小块，然后使用哈希函数对这些小块进行哈希计算，小块的哈希值被合并成更大的哈希值，逐级向上构建树状结构，直到最终生成一个代表整个文件的根哈希值；2）为了保证扫描 Merkle Tree 的正确性，Duggal 等将完美哈希指纹和完美哈希向量用于物理扫描算法中，提出了一种用于确定要迁移到云层的唯一数据块并估计文件迁移到云层的本地空间释放量的方法，为用户使用 Data Domain 的云层提供了便利。

4.1.3 指纹预取技术

通过指纹识别重复数据会随着数据量增大导致指纹不能全部存放在内存中，大部分需要存入磁盘[24]。而对于备份数据流中的一个数据块，从磁盘读取指纹进行对比的开销很大，导致整体重复数据删除性能降低。针对这一问题，Song 等[25]设计了一种基于文件相似度和数据局部性的指纹预取技术 FPP。FPP 通过获取两个文件的代表指纹判断两个文件的相似性，对于要进入存储系统的目标文件，若存在相似文件，则一次性预取相似文件的所有指纹，提高重复数据删除效率；若不存在相似文件，顺序预取上一次请求的磁盘地址后面的连续空间中存储的一部分指纹，利用数据局部性提高重复数据删除效率。为了保证空间局部性，所有确定存入的文件按数据流顺序存放在磁盘。

在 FPP 的基础上，Zhou 等[26]展开了更深入的研究，发现 FPP 的相似度识别模块是通过定长间隔采样的方式确定代表指纹，该方法对文件长度的变化异常敏感，文件的增删很容易导致采样点的变化，从而导致相似性检测的错误。Zhou 等采用可识别采样位置的算法 PAS[27]优化 FPP，并将基于哈希值

进行相似文件检测的方法 Simhash[28-30] 也加入了对比实验，二者都实现了比 FPP 更好的重复数据删除效率，且 PAS 算法的性能随着数据量和块大小的增大而提升。

Qin 等[31] 观察到渐近式采样的预取技术在面临没有历史信息的情况下表现出很差的性能[32]，他们开发了一个新的架构 PBCCF 来预取内容相关的备份中的指纹。PBCCF 建立在备份系统的客户端策略上，不同的客户端有不同的备份策略，而来自同一客户端的备份分布在多个容器当中，可以将容器使用情况作为判断备份相关性的标准。PBCCF 使用机器学习中的模式挖掘技术、基于密度的聚类技术和核密度估计将备份的容器使用分布拟合成二维正态分布周围的数据点，称为备份的特征，通过备份特征点的欧氏距离衡量备份的相关性。高度相关备份引用次数最多的几个容器中的指纹被预取到内存加速重复数据删除过程，很好地解决了渐进式采样的问题。

4.2 基于 sketch 的优化

考虑最常用的块级重复数据删除，内存中维护一个指纹列表，在目标块指纹与列表中指纹相同时认定为重复块，没有相同指纹时将目标块整个存入存储系统。对于这样的情况，有研究发现一些块即使指纹不同，其大部分内容也是相同的，若整块存入会导致数据削减率不高。在基于指纹识别重复块的效果不佳时，另一种识别重复内容的指标 sketch 出现了。sketch 用以判断两个块内容是否在很大程度上相似，传统的 SFSketch 方法如 Finesse[33] 通过多个哈希函数遍历块内容生成特征，再组合数个特征生成超级特征作为块的 sketch，然而这样的 sketch 生成方法面临误报和漏报的问题。Park 等[34] 基于机器学习中的 KNN[35] 聚类算法开发出动态聚类算法 DK-Clustering，并采用哈希学习[36] 方法为给定数据块生成 sketch，以保证内容相似的块被映射到汉明距离足够小的哈希值上，并将其命名为 DeepSketch 方法。带 sketch 的重复数据删除流程描述如下：对于请求的目标块，首先进行指纹对比，重复时添加到已有内容的索引；不重复时通过 NGT[37] 库的 ANN 搜索获取近似最临近参考块，将目标块与参考块进行 sketch 值计算和汉明距离比较，若判定为相似块，可进行逐字节比对仅将独特内容存入存储系统，再记录相似块对应关系，若 sketch 也表示没有足够相似的块再进行整块写入。虽然加入机器学习技术会导致 DeepSketch 吞吐量相比 Finesse 较慢，但其对于数据削减率的提升在强调存储空间的场景下具有不错的应用价值。

4.3 基于纠错码的优化

目前对 NVM 的研究面临一些瓶颈，比较严重的一个就是 NVM 非常强调性能问题，正如 LightDedup[19] 用非加密哈希＋逐字节比对替代复杂的哈希函数，在研究高性能且节能的加密 NVMM 时类似的问题更为明显：传统重复数据删除流程应用在 NVMM 上无法避免生成、存储和查询指纹需要的高计算、内存占用和索引查找开销，即使如今最先进的 DeWrite[38] 方案隐藏了哈希函数的计算延迟，但引入的巨大能量开销也不可忽视。

纠错码 ECC 提供对存储器位错误进行检测和纠错的能力，在缓存行从最后一级缓存中淘汰到 NVMM 时，内存控制器计算每条缓存线的 ECC，并将 ECC 与数据一起发送到特定位。更重要的是，ECC 可用于数据相似度识别[39]。若两个 cache line 的 ECC 值不同，则对应的缓存行内容也不同，否则其内容有很高的可能性是相同的。Du 等[40] 以 ECC 作为缓存行的特征开发了 ESD 方法，节省了哈希值计算的开销，对于 ECC 相同的缓存行，提取到内存进行逐字节比对以删除重复块；且由于最后一级缓存行极强的内容局部性，NVMM 中仅保留引用计数高的指纹并选择性进行重复数据删除，在大幅节省指纹存储开销的同时不损失太多性能。可惜不是所有的处理器都配备有带 ECC 的缓存，否则这种不需要计算指纹的方法的应用前景会更广阔。探索更多借助已有数据信息作为指纹的重复内容识别方法可能成为分层重复数据删除研究的重大突破口之一。

4.4 多层分级重复数据删除技术

单一的重复数据删除策略可能满足不了复杂的环境，因为一个存储系统工作负载和用户需求可能是时刻变化的，此时针对不同层的要求设计专门的重复数据删除服务也是一种理想的优化手段。Yin 等[41] 基于此开发了一种基于多层存储和 SLA 驱动的云存储系统重删框架 MUSE。MUSE 提出了 Dedup-SLA 作为后端云存储应用重删时服务供应商和客户间的专用协议，结合 I/O 性能和空间成本指标综合定义服务质量，将不同吞吐量和重删率的服务定义成不同级别的标准，根据用户需求确定每个层的 SLA 级别。然后 MUSE 将不同水平的重复数据删除服务分为 7 层，将系统运行过程分为多个监控周期，基于该层的 SLA 级别以及上一个周期的吞吐量和重删率动态确定该周期重删行为的升降级。

4.5 小结

对近十年重复内容识别方面的优化方法的总结如表 3 所列。

表 3 近十年重复内容识别优化技术

Table 3 Optimization techniques for deduplicate content identification in the past decade

时间	核心方法	成果	效果
2013	利用文件相似度和数据局部性预取指纹	FPP	提升重删速率
2014	用 PAS 和 Simhash 算法优化预取过程	PAS＋Simhash 优化 FPP	提升重删速率
2019	Merkle Tree＋完美哈希	Data Domain Cloud Tier	提升数据削减率
2019	获取数据块 Super-feature sketch 作为相似性特征	Finesse	提升数据削减率
2020	模式挖掘＋聚类＋核密度估计	PBCCF	提升重删速率
2021	自适应调整重删级别	MUSE	权衡成本和性能
2022	聚类＋哈希学习	DeepSketch	提升数据削减率
2023	纠错码代替指纹	ESD	提升重删速率
2023	非加密哈希＋逐字节比对	LightDedup	提升 NVM 性能

综合来看，重复内容识别作为提升整体系统性能最直接、最明显的步骤，近十年来的研究成果井喷式涌现，优化重删速率和提高数据削减率齐头并进，并逐渐呈现专门化的趋势，从寻找普适性的文件相似性特征或预取算法等转向针对具体的应用场景进行调优。为不同需求提高对应级别的重复数据删除策略，并随着 workload 的变化而调整将成为未来发展的主流趋势。

需要注意的是，重复数据删除技术固然具有节省空间的优点，但任何数据仅保留一份的存储方式难免会造成一定安全性和可靠性的风险，一旦唯一的数据块被污染或是在迁移过程中缺失，整个系统中甚至找不到供其恢复的备份，极大程度威胁着基于重复数据删除的分层存储技术的普及与发展。实际上，现在比较完备的基于重复数据删除的分层存储系统都带有日志恢复机制，例如 InftyDedup 和 Data Domain。另外，更多的选择是不过于极端地强调备份的唯一性，重复数据删除的初心是删除掉用户不需要的冗余数据，不要混淆重复数据和用于容错的备份之间的关系。用于容错的备份通常可以通过硬件隔离等安全措施存放在独立的存储区域，这方面不是基于重复数据删除的分层存储优化技术所关注的焦点问题，故不赘述。

5 针对数据迁移的优化技术

在确定好需要存储的非重复数据以及其对应的存储位置之后，就可以进行实际的数据迁移过程。数据迁移小可发生在缓存主存间的数据交换，迁移大可发生在多服务器多数据中心间的数据调度。关于数据替换策略的研究已有很长的历史，其关注的重点是在某个层容量接近上限的情况下将价值较低的数据替换为价值较高的数据；而数据迁移以更强的大局观统筹平衡整个系统各节点间的资源消耗，寻求全局的最优解。数据迁移步骤的主要流程如图 5 所示。

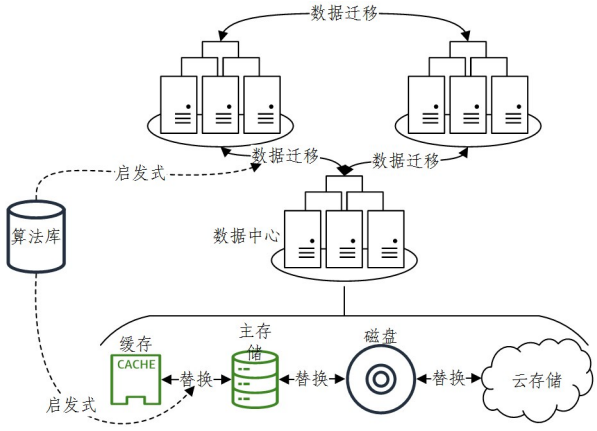


图 5 数据迁移流程

Fig. 5 Process of data migration

5.1 优化数据替换策略

5.1.1 FIFO 策略

FIFO 意为 First In First Out，即对于要发生替换的存储设备，优先替换最早进入该设备的数据块。该算法实现简单，只需把调入存储设备的页面根据先后次序链接成队列，设置一个指针总指向最早的页面。但该算法与进程实际运行时的规律不符，一些经常被访问的数据在 FIFO 算法下也面临被淘汰的风险。

5.1.2 LRU 策略

LRU 意为 Last Recently Used，其利用了局部性原理，认为近段时间不被访问的块在未来也不大可能被访问，故每次替换掉存储设备中最长时间未被使用的块。LRU 一定程度上保护了近段时间被使用过的块，且在理论上证明了在最坏情况下具有优秀的性能[42]。但 LRU 仍存在新的问题：有些块在之前被频繁使用，但间隔了一段时间未被访问，会导致其处于 LRU 列表较危险的位置。

5.1.3 LFU 策略

LFU 意为 Last Frequently Used，其基本思想是，如果一个数据在一段时间很少被访问到，那么可以认为它在将来被访问的可能性也很小，故每次替换存储设备中访问频率最低的块。如果多个块的访问频率相同，则按照 LRU 的思想淘汰最长时间未被访问的块。

5.1.4 ARC 策略

ARC[43]使用 T1，T2，B1，B2 这 4 个列表，T1+T2 不超过存储设备容量。T1 采用 LRU 策略，按访问时间排序，但只储存访问一次的数据，当相同数据被第二次访问时，将该数据从 T1 移到 T2，T2 按照 LFU 策略，按访问频率排序。从 T1 和 T2 替换出来的数据分别进入 B1 和 B2。若后续访问到 B1 中的数据，说明 LRU 策略占优，T1 长度+1，T2 长度−1；若访问到 B2 中的数据，说明 LFU 策略占优，T1 长度−1，T2 长度+1。ARC 算法综合考虑了数据块的访问频率和时间，大大提高了缓存命中率。

5.1.5 LIRS 策略

LIRS[44]也是为了解决单一 LRU 算法的局限性而被开发出来的综合考虑频率和时间间隔的算法。LIRS 采用 IRR 描述一个数据块上两次被访问之间访问的其他独特数据块的个数；采用 Recency 描述一个数据块从上一次访问到当前时间访问的独特数据块个数。简单来说，每次替换会优先选择 IRR 值最大的项为目标，在 IRR 相同的情况下，则优先选择 Recency 值更大的数据块。

5.1.6 MQ 策略

MQ[45]，即 Multi-Queue，其使用 m 个按生存周期长度升序排列的 LRU 队列：Q_0-Q_{m-1}。还维护一个 FIFO 队列 Q_{out} 来存放那些一段时间内被淘汰的块的访问频度 f。

块 b 缓存命中时，首先从当前 LRU 队列中移除 b，然后根据 b 的当前访问频度 f，计算出新的存储队列号 k，将 b 从当前队列移到 Q_k 末尾，使得访问频度高的块处在更安全的位置。块 b 缓存未命中时，MQ 算法将最低非空队列的 LRU 块 c 淘汰到 Q_{out} 并在必要时触发 Q_{out} 的替换，再计算新块 b 的存放位置。若 b 之前在 Q_{out} 中，移动 b 并将之前保留的 $f+1$ 作为 b 的 f 返回，否则 b 的 f 置 1。这样长时间未被访问的块会被及时淘汰。

MQ 还对块应用了 DEMOTE[46]降级操作，每个队列有自己的生存周期，每个块进入队列时都有一个到期时间，若这个块在到期时间内未被访问，则移到下一级队列的 MRU 位置并重置到期时间，这样能保证热块在高级队列中至少存在一定时间，但长时间未被访问依旧会有被淘汰的风险。

MQ 虽然能提高二级缓存的命中率，但仅适用于缓存失效开销稳定的单一型存储。面对混合存储服务器连接多种网络存储设备和本地存储设备的环境时，由于不同存储设备的 I/O 性能差异较大，其缓存失效开销有别于传统的直连存储，

因此在保证命中率的同时,还应通过降低失效开销来提高缓存系统性能[47],优先保存访问频度高且失效开销大的数据,存在一定的优化空间。

5.1.7 D-LRU 和 D-ARC 策略

无论是 LRU,LFU 还是更好的 ARC 算法,在面临重复数据删除系统时都存在一个同样的问题:在不考虑重复数据的情况下,访问一个地址造成的替换可直接对应于数据块的移动;但在重复数据删除系统中往往存在多个地址映射到相同数据块的情况,若地址的访问直接引起数据块的替换则可能导致其他引用该数据块的地址也受到影响。

基于此,Li 等[48]设计了一种 inline 的重复数据删除方案 CacheDedup 来解决替换算法的落后问题。CacheDedup 设计了两种新的缓存替换算法 D-LRU 和 D-ARC。D-LRU 算法采用两个缓存 M 和 D,M 中存源地址及其对应的指纹,D 中存放可通过指纹索引的实际数据块,LRU 替换同时发生在两个缓存中,可通过 M 中的缓存项恢复可能已从 D 中淘汰的对应缓存项。

D-ARC 算法与传统的用于 cache 替换的 ARC 算法的不同之处有以下 3 点:1)4 个列表都按 LRU 策略进行淘汰,且 T1+T2 的总容量不固定,而是随着重复数据数量而变化。对于存储 C 个条目的数据缓存,若无重复块,则元数据缓存将最多保留 C 个源地址,每个源地址映射到 D 中的一个唯一块;在存在重复块的情况下,D-ARC 允许 T1 和 T2 的总条目增加到 $C+X$,以便存储 X 个具有重复块的源地址。2)在 B1 和 B2 中命中的源地址将被存回 T2 并造成 T1 和 T2 的长度变化,对于 B1 和 B2 中替换出的源地址-指纹对,新增一个额外 LRU 列表 B3 来保存它们,B3 本质上是利用 T1+T2 的剩余空间,即 $T1+T2+B3<C+X$。当访问的源地址在 B3 中命中时,移回 T1 但不改变 T1 和 T2 的长度。3)当需要在数据缓存中替换时,D-ARC 仅淘汰在 $T1\cup T2$ 中没有映射的项。若没有这样的数据项,则在 T1 和 T2 中淘汰源地址和指纹,直到能找到无映射的数据块,这个过程中最多有 $X+1$ 项被从 $T1\cup T2$ 中替换。经实验证明,CacheDedup 大大提高了 I/O 性能和闪存耐用性,为基于重复数据删除的分层存储系统的替换算法设计做出了突出贡献。

5.2 优化迁移算法

一切准备就绪,数据可以开始在不同层之间进行迁移。Cao 等[49]于 2020 年开发了一个基于快慢分层的重复数据删除文件系统 TDDFS,其关于指纹生成、指纹对比等细节设计与传统的重复数据删除并没有太大区别,关键设计点在于 TDDFS 规定了 3 种文件类型及针对它们的迁移操作:新创建的文件、经过重复数据删除的文件、重新加载的文件。所有文件的元数据都存储在快层,新创建文件和重新加载文件存储在快层,经过重复数据删除的文件存储在慢层,新创建文件以整个文件为单位存储,而经过重复数据删除和重新加载的文件以块为单位存储。当快层文件超过最大容量时,需要将不活跃的文件迁移到慢层,若选中的是新创建文件,触发重复数据删除操作,则将文件分块比对,仅将慢层没有的块存入;若选中重新加载文件且文件未经修改,则不进行重复数据删除直接进入慢层,否则按新创建文件处理。经过重复数据删除的文件若收到访问请求,则触发重新加载操作,将慢层的该文件迁移回快层,但 TDDFS 不像大多数重复数据删除系统一样迁移整个文件,而是只迁移快层中重新加载文件中没有的块,同时用元数据的 modification 字段标识该块重新加载后是否被更改。TDDFS 在保证高重复数据删除性能的情况下降低了存储成本,实现了高性价比。

Data Domain[20]针对本地层和云层的迁移操作也进行了设计,在估计完将数据迁移到云层可释放的本地空间后,针对用户第一次购买许可证之后的迁移操作采取批量播种技术,即通过完美哈希向量和物理扫描技术按宽度优先遍历 Merkle Tree,将大量数据从本地层迁移到几乎为空的云层。由于此时云层几乎没有用户数据,可以节省重复数据删除流程。对于之后的文件迁移请求,Data Domain 仍采取深度遍历 Merkle Tree 定位迁移块并通过指纹对比进行重复数据删除后进行实际数据移动操作。

在分布式环境普及的当下,迁移不仅涉及不同层级间的数据,还包括同级别层不同设备间文件重映射等造成的数据移动。Kisous 等[50]将分布式重复数据删除系统的一般迁移问题表述为一个优化问题,其目标是最小化系统的大小,同时将确保存储负载在系统卷之间均匀分布以及迁移所需的网络流量不超过每个负载分配的上限作为问题的限制。针对这一优化问题,Kisous 等提出了 3 种生成有效迁移计划的算法,每种算法都基于不同的方法,并在计算时间和迁移效率之间进行了不同的权衡。贪心算法是在 Harnik 等的工作[51]上进行的改良,其本质上是一个循环遍历的过程,将整个迁移过程分为多个重复阶段,每个阶段寻求在满足流量和负载均衡要求的情况下最小化存储空间的迁移方案,其优化效果有限,但其优势在于运行速度极快。ILP 算法基于 GoSeed[52]算法进行了更一般性的扩展,将数学上的线性规划方法运用在迁移数据量的优化上,采用 Gurobi 优化器[53]实现了比贪心算法更明显的效果,但其复杂度很高且不能保证一定找到最优解。分层聚类算法将相似文件划分到一个集群,保证集群个数等于服务器个数,然后将每个集群中的文件映射到一个服务器中,迁移与初始状态不同的文件和块,这样每个服务器中的文件共用最多的块,使用最少的空间。分层聚类算法兼顾了之前两种算法的优点,在优化效果与 ILP 算法相当的前提下节省了比 ILP 最高一个数量级的时间成本。

5.3 小结

对近十年数据迁移方面的优化方法的总结如表 4 所列。

表 4 近十年数据迁移优化技术

Table 4 Optimization techniques for data migration in the past decade

时间	核心思想	优化成果	优化目的
2016	D-LRU+D-ARC	CacheDedup	提升缓存命中率
2019	批量播种与重删的配合使用	Data Domain Cloud Tier	提升迁移速率
2020	为不同类型文件设计专门化迁移策略	TDDFS	提升数据削减率和迁移速率
2022	对比实现贪心+ILP+分层聚类算法	贪心+ILP+分层聚类算法	综合考虑流量、存储空间和负载均衡

不难发现，随着时间的推移，对于数据迁移操作的优化已不再拘泥于极致地追求节省存储空间和加速系统性能，而是更多地考虑到资源合理利用的问题。随着国家“东数西算”工程的部署，数据的迁移将更多地发生在多服务器、多数据中心之间，如何保证各节点间的负载均衡，不超出网络带宽限制，最大化利用每个数据中心的系统性能将成为一个长期课题。

6 未来发展

当今时代数据的爆炸性增长以及云存储相关技术的普及使得分层存储系统和重复数据删除技术展现出越来越大的使用价值。

基于重复数据删除的分层存储优化技术的研究成果呈现出逐年递增的趋势，彰显了其关注度和重要性的稳步提升，未来的研究方向应该集中于以下几个方面。

1)探索人工智能技术与基于重复数据删除的分层存储的高效融合。目前 AI 在存储领域的应用已经涉及方方面面[54]，对于基于重复数据删除的分层存储来说，其 3 个关键步骤都有一定的优化空间。例如使用机器学习分类算法帮助数据分层；利用聚类算法实现多节点间数据传输的资源平衡；利用数据挖掘和模式识别增强数据的相似性检测等。这些研究方向的关键在于训练样本的质量以及训练速度和精度的权衡。考虑到算力的问题，在大型分布式系统上部署会有更好的预期效果。

2)设计自适应的重复数据删除算法。目前已有研究为平衡 I/O 性能和空间成本设计了不同质量的重复数据删除方法。未来可以借鉴这一思路，针对不同存储层的要求设计更细化的指标，例如对于空间敏感型的存储介质采用更强级别的重复数据删除，对于性能敏感型的存储介质适当降低重复数据删除的删除率来换取带宽的提升，并且能在工作场景变化的情况下动态切换不同性能的标准。

同时，在某些情况下，对于首次向一个几乎为空的层进行数据迁移时，重复内容识别的命中率极低，此时运行复杂的哈希计算、查找等收益远弥补不了开销。理想的重复数据删除算法应能感知这类特殊情况，适时暂停服务以提供最优的性能。

3)设计更好的效用值函数。数据替换策略方面，常用的算法都是以命中率作为评判启发式效果的标准，这是基于快速层未命中的数据需要到访问时间更长的慢速层获取这一特点。但在混合型存储介质普及的当下，同一存储层下不同介质的访问延迟也有差距，快速层缓存的数据应该向未命中开销更大的介质倾斜，此时应该整合命中率和失效开销设计新的启发式函数。数据迁移方面，从多设备向更大范围的多数据中心扩展，要综合考虑数据特性、网络特性、数据中心特性等多方面因素，既追求全域重复数据删除的最高存储空间利用率，又尽可能保证数据及时响应、安全可用。

4)寻找表征能力更强的特征。受 sketch 和 ECC 的启发，相较于指纹，sketch 能提供更高的重复数据缩减率但需要更细致的计算处理，ECC 免除了哈希计算的开销但粒度固定且不在所有系统中普及。能否找到一个结合二者优点的特征从根本上取代哈希值指纹是决定重复数据删除性能是否有开创性突破的关键。

总的来说，基于重复数据删除的分层存储具有非常广阔的应用前景。随着研究的深入，理想中的兼顾安全和高性价比的存储系统一定会被开发出来，在这个大数据时代切实解决用户的问题，满足用户的需求，优化用户的体验。

结束语 本文以当前存储领域面临的问题为引子，聚焦于基于重复数据删除的分层存储技术，先从相关概念、分类方法、主要流程等方面介绍了其原理，再从三大关键步骤入手深入分析总结了各类基于重复数据删除的分层存储优化技术的创新思路和技术挑战，并对其潜在风险进行简要讨论，最后对该技术的未来发展趋势进行了展望。

参 考 文 献

[1] 国际数据中心 IDC[EB/OL]. https://www.idc.com/.

[2] IBM 数据生命周期管理[EB/OL]. https://www.ibm.com/cn-zh/topics/data-lifecycle-management.

[3] XIE P. Survey on Data Deduplication Techniques for Storage System. [J]. Computer Science, 2014, 41(1): 22-30, 42.

[4] FU Y, XIAO N, LIU F. Research and Development on Key Techniques of Data Deduplication[J]. Journal of Computer Research & Development, 2012, 49(1): 12-20.

[5] LEE T, MONGA S K, MIN C W, et al. Memtis: Efficient Memory Tiering with Dynamic Page Classification and Page Size Determination[C]// ACM SIGOPS 29th Symposium on Operating Systems Principles. ACM, New York, NY, USA, 2023.

[6] HILDEBRAND M, KHAN J, TRIKA S, et al. AutoTM: Automatic Tensor Movement in Heterogeneous Memory Systems using Integer Linear Programming[C]// Architectural Support for Programming Languages and Operating Systems. ACM, 2020.

[7] Amazon Web Services. Amazon s3 price[EB/OL]. https://www.amazonaws.cn/s3/pricing/.

[8] Microsoft Azure. Storage Price [EB/OL]. https://azure.microsoft.com/zh-cn/pricing/details/storage/blobs/#pricing.

[9] 百度云对象存储 BOS[EB/OL]. https://cloud.baidu.com/product/bos.html.

[10] 阿里云对象存储 OSS[EB/OL]. https://www.aliyun.com/product/oss.

[11] 腾讯云对象存储 COS[EB/OL]. https://cloud.tencent.com/product/cos.

[12] KOTLARSKA I, JACKOWSKI A, LICHOTA K, et al. InftyDedup: scalable and cost-effective cloud tiering with deduplication[C]// Proceedings of the 21st USENIX Conference on File and Storage Technologies. 2023.

[13] YANG Z Y, WANG Y F, BHAMIN I, et al. EAD: elasticity aware deduplication manager for datacenters with multi-tier storage systems[J]. Cluster Computing, 2018, 21(3): 1561-1579.

[14] WANG H, ZHANG J W, HUANG P, et al. Cache What You

Need to Cache:Reducing Write Traffic in Cloud Cache via "One-Time-Access-Exclusion" Policy[J]. ACM Transactions on Storage,2020,16(3):1-24.

[15] ETHEM A. Introduction to Machine Learning[J]. MIT Press, Cambridge,MA,2014.

[16] LEO B,FRIEDMAN J H,OLSHEN R A,et al. Classification and Regression Trees[J]. Biometrics,1984,40(3),358.

[17] XIA W,JIANG H,FENG D,et al. A comprehensive study of the past,present,and future of data deduplication[J]. Proceedings of the IEEE,2016,104(9):1681-1710.

[18] WANG C D,WEI Q S,YANG J,et al. Nv-dedup:Highperformance inline deduplication for non-volatile memory[J]. IEEE Transactions on Computers,2017,67(5):658-671.

[19] QIU J S,PAN Y Q,XIA W,et al. Light-Dedup:A Light-weight Inline Deduplication Framework for Non-Volatile Memory File Systems[C]// USENIX Annual Technical Conference. 2023:101-116.

[20] BOTELHO F C,GARG N,SHILANE P N,et al. Memory efficient sanitization of a deduplicated storage system. 中国专利:US9317218[P],2016.04.19.

[21] DUGGAL A,JENKINS F,SHILANE P,et al. Data domain cloud tier:backup here,backup there,deduplicated everywhere![C]// USENIX Annual Technical Conference. 2019.

[22] ZHU B,LI K,PATTERSON H. Avoiding the disk bottleneck in the Data Domain deduplication file system[C]// 6th USENIX Conference on File and Storage Technologies. 2008.

[23] MERKLE RALPH C. Digital signature system and method based on a conventional encryption function,US7967587A[P]. 1987.07.30.

[24] ESHGHI K,LILLIBRIDGE M,WILCOCK L,et al. Jumbo store:Providing efficient incremental upload and versioning for a utility rendering service[C]// 5th USENIX Conference on File and Storage Technologies. 2007.

[25] SONG L S,DENG Y H,XIE J J. Exploiting Fingerprint Prefetching to Improve the Performance of Data Deduplication[C]// IEEE International Conference on High Performance Computing & Communications & IEEE International Conference on Embedded & Ubiquitous Computing. 2013.

[26] ZHOU Y T,DENG Y H,XIE J J. Leverage similarity and locality to enhance fingerprint prefetching of data deduplication[C]// IEEE International Conference on Parallel and Distributed Systems. 2014.

[27] ZHOU Y T,DENG Y H,CHEN X G,et al. Identifying file similarity in large data sets by modulo file length[C]// Algorithms and Architectures for Parallel Processing. 2014.

[28] MANKU G S,JAIN A,DAS S A. Detecting near-duplicates for web crawling[C]// International Conference on World Wide Web. ACM,2007.

[29] CHARIKAR M S. Similarity estimation techniques from rounding algorithms[C]// Thiry-fourth Acm Symposium on Theory of Computing. ACM,2002:380-388.

[30] INDYK P,MOTWANI R. Approximate nearest neighbors:towards removing the curse of dimensionality[C]// Proceedings of the 30th ACM Symposium on Theory of Computing(STOC'98) 1998:604-613.

[31] QIN Y B,ZHANG X B,DAVID J. PBCCF:Accelerated Deduplication by Prefetching Backup Content Correlated Fingerprints[C]// 2020 IEEE 38th International Conference on Computer Design. 2020.

[32] GUO F,EFSTATHOPOULOS P. Building a high-performance deduplication system[C]// 2011 USENIX Annual Technical Conference. USENIX Association,2011.

[33] ZHANG Y C,XIA W,FENG D,et al. Finesse:Fine-grained feature locality based fast resemblance detection for postdeduplication delta compression[C]// USENIX FAST. 2019.

[34] PARK J,KIM J,KIM Y,et al. DeepSketch:A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression[C]// 20th USENIX Conference on File and Storage Technologies(FAST 22). 2022:247-264.

[35] LLOYD S P. Least squares quantization in PCM [J]. IEEE Trans.,1982,28(2):129-137.

[36] SU S P,ZHANG C,HAN K,et al. Greedy hash:Towards fast optimization for accurate hash coding in cnn[C]// NIPS. 2018:806-815.

[37] Yahoo! Japan Corp. Neighborhood graph and tree for indexing high-dimensional data[EB/OL]. https://github.com/yahoojapan/NGT.

[38] ZUO P F,HUA Y,ZHAO M,et al. Improving the Performance and Endurance of Encrypted Non-Volatile Main Memory through Deduplicating Writes[C]// 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture(MICRO). ACM,2018.

[39] JAULMES L,MORETO M,VALERO M,et al. A Vulnerability Factor for ECC-protected Memory[C]// 2019 IEEE 25th International Symposium on On-Line Testing And Robust System Design(IOLTS). IEEE,2019.

[40] DU C F,WU S Z,WU J P,et al. ESD:An ECC-assisted and Selective Deduplication for Encrypted Non-Volatile Main Memory[C]// 2023 IEEE International Symposium on High-Performance Computer Architecture. 2023.

[41] YIN J W,TANG Y,DENG S G,et al. MUSE:A Multi-Tierd and SLA-Driven Deduplication Framework for Cloud Storage Systems[J]. IEEE Transactions on Computers,2021,70(5):759-774.

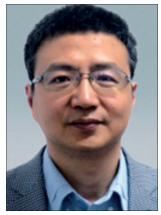
[42] SLEATOR D D,Tarjan R E. Amortized efficiency of list update paging rules[J]. Communications of the ACM,1985,28(2):202-208.

[43] MEGIDDO N. ARC:A self-tuning,low overhead Replacement cache[C]// USENIX File and Storaqe Technologies Conference(FAST'03). 2003.

[44] JIANG S. LIRS:An Efficient Low Inter-reference Recency Set Replacement Policy to Improve Buffer Cache Performance[C]// Proceedings of the International Conference on Measurements and Modeling of Computer Systems,2002.

[45] ZHOU Y Y, PHILBIN J, LI K. The multi-queue replacement algorithm for second level buffer caches[C]// Proceedings of the USENIX Annual Technical Conference. CA, USA, 2002: 91-104.
[46] WILKES T M W J. My cache or yours? Making storage more exclusive[C]// Proceedings of the General Track: 2002 USENIX Annual Technical Conference. 2002.
[47] XIAO N, ZHAO Y J, LIU F, et al. Dual queues cache replacement algorithm based on sequentiality detection[J]. Science China(Information Sciences), 2012, 55(1): 191-199.
[48] LI W J, GREGORY J B, JUAN R, et al. CacheDedup: in-line deduplication for flash caching[C]// Proceedings of the 14th Usenix Conference on File and Storage Technologies. 2016.
[49] CAO Z C, WEN H, GE X Z, et al. TDDFS A Tier-Aware Data Deduplication-Based File System [J]. ACM Transactions on Storage, 2019, 15(1): 4.
[50] KISOUS R, KOLIKANT A, DUGGAL A, et al. The what, The from, and The to: The Migration Games in Deduplicated Systems[J]. ACM Transactions on Storage, 2022, 18(4): 1-29.
[51] HARNIK D, HERSHCOVITCH M, SHATSKY Y, et al. Sketching volume capacities in deduplicated storage [C] // 17th USENIX Conference on File and Storage Technologies. 2019.
[52] NACHMAN A, SHEINVALD S, KOLIKANT A, et al. GoSeed: Optimal seeding plan for deduplicated storage[J]. ACM Transactions on Storage, 2021, 17(3): 1-28.
[53] Gurobi[EB/OL]. https://www.gurobi.com/.
[54] LIU Y, WANG H, ZHOU K, et al. A survey on AI for storage [J]. CCF Transactions on High Performance Computing, 2022, 4(3): 233-264.

YAO Zilu, born in 2001, postgraduate, is a member of CCF (No. P8042G). His main research interests include tiered storage and deduplication and so on.



XIAO Nong, born in 1969, Ph.D, professor, doctoral supervisor. His main research interests include large-scale storage system, cloud computing and network computing, and computer architecture.

(责任编辑:何杨)

2024 CCF 会士名单公布

经 CCF 会士评选委员会评选,14 位 CCF 会员当选 2024 CCF 会士。谨向新当选会士表示祝贺!

会士是会员在 CCF 的最高学术荣誉,用于表彰在计算机领域有卓越成就或为 CCF 做出突出贡献的 CCF 会员。

2024 年当选的会士名单如下(按姓氏拼音排序):

陈海波	教 授	上海交通大学
陈 为	教 授	浙江大学
慈林林	研究员	火箭军研究院
崔 斌	教 授	北京大学
高 阳	教 授	南京大学
管海兵	教 授	上海交通大学
姜育刚	教 授	复旦大学
李克秋	教 授	天津大学
刘淮松	研究员	中国电子科技集团有限公司
孙晓明	研究员	中国科学院计算技术研究所
王 泉	教 授	西安电子科技大学
吴 杰	教 授	中国电信云计算研究院
徐常胜	研究员	中国科学院自动化研究所
仲 盛	教 授	南京大学

据 CCF 微信公众号