

**研 究 生 ( 数 据 中 心 技 术 ) 课 程 论 文 ( 报 告 )**

**题目：数据中心重复数据删除的性能优化**

|  |  |
| --- | --- |
| 学 号 | M202073162 |
| 姓 名 | 严伟杰 |
| 专 业 | 电子信息 |
| 课程指导教师 | **施展、童薇** |
| 院（系、所） | **武汉光电国家研究中心** |

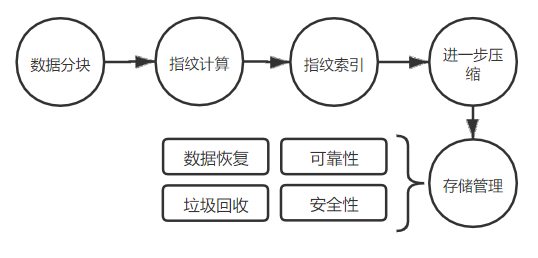
2020年12月10日

## 1 引言

随着互联网时代的到来，全球数据量呈现指数式增长。根据IDC的报告，2020年全球生成的数据量将超过44ZB[1]。随着数据中心中存储的数据量的增加，冗余数据的比例也在不断增加。根据Microsoft[2][3]和EMC[4]的研究，它们的数据中心中分别约有50%和85%的数据是冗余的重复数据。IDC最近的一项研究显示，接近80%接受调查的公司正在探索其存储系统中的重复数据删除技术以减少冗余数据[5]。如何有效地管理数据，提高数据存储效率并减低存储成本成为了当前学术研究的热点之一。为了有效地管理大量冗余数据，重复数据删除技术已在数据中心得到广泛应用[6]。

重复数据删除技术是一类无损的数据压缩技术。它不仅可以通过消除重复数据来减小存储空间，还可以减小低带宽网络环境中冗余数据的传输，从而提高传输性能。重复数据删除技术通常可以分为两类：离线重复数据删除和在线重复数据删除。离线重复数据删除是在数据写入存储设备后，在后台消除存储设备中的冗余数据。而在线重复数据删除则是实时地边存储数据边进行冗余数据的消除。离线重复数据删除会占据更多的存储空间，并且不会减少数据的写操作。因此，在线重复数据删除具有更好的研究价值。

典型的在线重复数据删除系统遵循着数据分块，指纹计算，指纹索引，进一步压缩和存储管理的工作流程[6]。重复数据删除系统中的存储管理可以分为几类，包括数据恢复，垃圾回收，可靠性，安全性等，如图1所示。数据分块是利用数据分块算法将数据流分成许多大小不等的数据块。相比于早期的文件级的重复数据删除技术，数据块级的重复数据删除技术可以实现更高的去重率。指纹计算是利用哈希算法计算出每个数据块的指纹。数据块指纹作为数据块的唯一标识，系统可以通过比较数据块指纹是否相同来判断数据块是否重复。指纹索引是重复数据删除系统的关键阶段。系统为已经存储的数据块指纹建立索引，通过查询索引来判断数据是否重复。进一步压缩阶段指的是将那些非重复但是非常相似的数据块进行压缩存储，即差量压缩。



**图1 重复数据删除技术的工作流程**

重复数据删除的性能对于数据中心至关重要，而重复数据删除系统中涉及许多计算和磁盘索引等严重影响I/O性能的操作，因此需要采用优化技术来提高重复数据删除系统的吞吐量。近年来，不同研究人员分别从指纹索引、差量压缩和数据恢复对重复数据删除系统进行了优化，取得了显著的效果。

## 2 基于指纹索引的优化

论文发表于MSST 19，论文名称：LIPA: A Learning-based Indexing and Prefetching Approach for Data Deduplication[7]

### 2.1 论文背景

在基于数据块的重复数据删除系统中，一个非常重大的挑战是如何构建有效的指纹索引以帮助识别重复的数据块。当数据量很大时，在RAM中保持非常大的指纹索引是不切实际的。但是如果将索引放在磁盘里，频繁的磁盘读取又会严重影响I/O性能。这个问题被称为块查找磁盘瓶颈问题[10]。例如，对于1PB的唯一数据集，假设平均块大小为8KB，它将生成大约2.5TB SHA-1指纹（即每个块160位）。指纹索引结构在查找等待时间和重复数据删除率方面会严重影响重复数据删除性能，因为每次只能访问这些指纹的一部分。

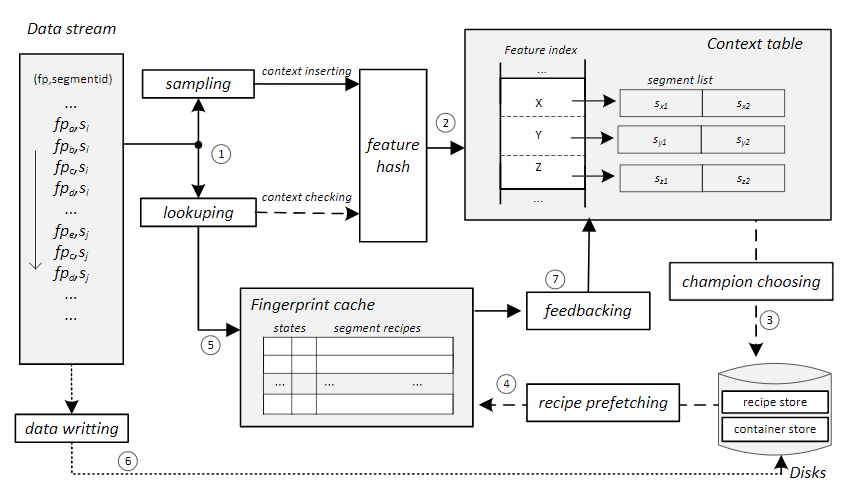
针对块查找磁盘瓶颈问题，很多研究机构已经提出了一些解决方案。Sparse indexing[11]通过对内存中的数据块指纹索引进行采样来提高重复数据删除系统的内存利用率，从而将内存使用率降低到一半以下。Extreme Binning[12]通过利用文件相似性来实现每个文件的单个磁盘索引访问以进行块查找，从而提高了重复数据删除的可伸缩性。

经过研究发现：1.对于基于采样的指纹索引，采样率与内存开销成比例并进一步直接影响重复数据删除率。2.采样的指纹与数据段之间的映射关系是静态的，缺乏自适应反馈机制来调整映射关系以反映数据流的动态。

### 2.2 方法实现

论文提出了利用一种简单的强化学习方法建立动态预取数据段的索引，简称LIPA。论文采用了一种称为K臂赌博机的特定强化学习模型[13]，让索引可以动态地选择更优的数据段进行预取，从而减少磁盘读取的次数，提高I/O的性能。

与传统的指纹缓存的方法相比，LIPA增加了一个核心组件：上下文表。上下文表用于存储重复数据删除过程中指纹查找的运行时上下文信息。对于即将到来的数据段，系统将数据段采样的特征映射到上下文表中，并建立其特征和数据段的映射关系。上下文表具有两个重要的属性：段的分数和段的追随者。分数表示随着时间的推移命中对该数据段进行查找的奖励。它由数据段在指纹缓存中的平均命中次数表示。追随者表示当前数据段被选择为冠军后将预取到缓存中的连续数据段的数量。追随者的数量由预取的效果来动态决定。当从指纹缓存中获取反馈时，它们会被更新。LIPA的指纹缓存存储着数据段的指纹信息和命中次数。从系统上下文表中选择冠军数据段时，该冠军数据段及其一些追随者数据段将被预取到指纹缓存中，同时某些数据段将被驱除出指纹缓存。当数据段被驱逐出指纹缓存后，它会将它记录的命中次数反馈到上下文表中，作为该数据段的分数信息。



**图2 LIPA的架构图**

LIPA的架构图如图2所示。它的流程如下：

1. 对输入的数据块指纹进行采样获取数据段特征。
2. 将数据段特征插入上下文表，构造特征和数据段之间的映射。
3. 从当前的上下文表中利用算法选出冠军数据段。
4. 将冠军数据段和后续的数据段预取到指纹缓存中。
5. 在指纹缓存中检查对数据段的每个数据块是否重复。
6. 非重复的数据块被写回磁盘。
7. 如果有数据段从指纹缓存中被驱除出去，则将奖励进行反馈，更新上下文表。

LIPA的冠军选择策略主要是最近选择和贪心选择两种。最近选择策略总是选择最新的数据段作为冠军，而贪心选择策略是指每次有的概率选择具有最高分数的数据段，有的概率随机选择。这是强化学习算法中探索与利用的平衡。

### 2.3 实验评估

为了评估LIPA的效果，作者针对Linux Kernel、Vmdk、Fslhomes和Macos四个数据集，将LIPA与传统的Sparse Indexing的方法进行了对比实验测试。实验基于Destor[14]的实验测试平台，采用TTTD和CDS作为分块和分段算法，利用SHA-1计算数据块的指纹。实验的采样方法为最小采样。实验结果如图3所示。

|  |  |
| --- | --- |
| C:\Users\ASUS\AppData\Roaming\Tencent\Users\1332765311\TIM\WinTemp\RichOle\}5CPK7)3AMHPG{9D`EFU`FA.png | C:\Users\ASUS\AppData\Roaming\Tencent\Users\1332765311\TIM\WinTemp\RichOle\5P5@TUZHU9JX308R@Z2`TNH.png |

**图3 去重率和吞吐量实验结果图**

由实验结果可知，LIPA不仅可以提高数据去重率，在部分数据集上还可以极大提升I/O性能。

### 2.4 论文总结

论文的最大贡献在于首次在重复数据删除技术中加入机器学习的方法，提出了一种基于强化学习的指纹索引检测与预取技术以解决磁盘瓶颈问题。系统通过反馈机制更新了特征及其对应数据段之间的关联关系并动态调整了缓存机制以提高重复数据删除系统的性能。实验结果表明基于学习的方法只需要很少的内存开销即可存储索引，同时与以前的方法相比，其重复数据删除率和吞吐量甚至更高。系统可以根据不同的用户模式动态地获得更好的性能。

## 3 基于差量压缩的优化

论文发表于FAST 19，论文名称：Finesse: Fine-Grained Feature Locality based Fast Resemblance Detection for Post-Deduplication Delta Compression[8]

### 3.1 论文背景

在存储系统中，差量压缩通常用作重复数据删除技术的补充技术，因为它能够消除非重复但高度相似的数据块之间的冗余。例如，如果数据块A2与数据块A1相似，那么差量压缩方法可以仅存储或传输A2和A1之间的差异和映射关系，从而删除冗余数据以提高存储空间效率和网络带宽效率。差量压缩的一般工作流程是：①计算数据块的相似度，即计算特征和特征分组。②检查相似的数据块。③对两个相似的块进行编码，即计算它们的差量。

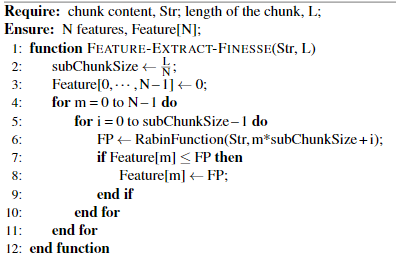
当前，N-transform SF[15]是最广泛使用的计算数据块相似度以检测差量压缩候选数据块的方法。N-transform SF的具体算法是：利用Rabin指纹在数据块上逐字节计算，然后对每个Rabin指纹分别线性变换N次以计算N维哈希值集。最后，从N个维度中的每个维度中选取N个最大值作为特征。由于算法需要进行N次线性变换，所以计算非常耗时。

### 3.2 方法实现

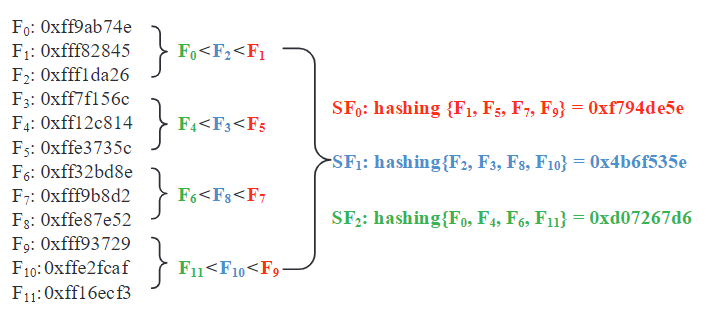
论文观察到相似块之间也存在细粒度的局部性。数据块的相应子区域（子块）及其特征在相似的数据块中也以很高的概率出现在相同的顺序中，这被称为特征局部性。基于以上的事实，论文提出了Finesse，这是一种基于特征局部性的细粒度快速重相似度检测方法，该方法将每个块划分为几个固定大小的子块，分别计算这些子块中的特征，然后将特征分组为超特征，从而简化了相似度计算。

Finesse的具体实现包括特征提取和特征分组。特征提取：为了利用相似块的细粒度特征局部性来提取特征，Finesse首先将数据块划分为几个固定大小的子块，然后根据数据内容的Rabin指纹在每个子块上计算特征，此处与传统的N-transform SF方法相同。由于基于内容分块算法会将数据块分成大小不一的子块，从而使计算变得非常复杂。因此论文采用固定大小分块的方法划分子块。特征提取算法如图4所示。

特征分组：Finesse首先将子块及其对应的特征划分为几个相同大小的连续集合，然后将最大的特征（每个集合中具有最大哈希值）分组以构成第一个超特征，将集合的第二个最大特征分组以形成第二个超特征，依此类推。特征分组如图5所示。图5中Finesse首先将块划分为12个子块，以提取12个特征F0 ... F11并分为四组。之后从每个组中选择最大值、次大值组成第一个超特征和第二个超特征，以此类推。



**图4 Finesse特征提取算法**



**图5 Finesse特征分组**

N-transform SF需要至少3xN次操作，而Finesse只需要一次分支操作，因此大大加快了系统的性能。Finesse的局限性在于它无法检测到大小相差很大的相似块。因为如果两个相似块的大小相差很大，那么它们通过划分子块并分组所得到的特征将完全不同。

### 3.3 实验评估

实验在Destor平台上分别用3.4GHz的Intel i7-4770处理器和2.4GHz的Intel E5-2620处理器进行对比测试。系统分块大小为8KB，指纹算法为SHA-1。实验采用第一个匹配的块作为差量压缩的基本块。实验数据集分别为WEB,TAR,RDB,SYN,VMA,VMB。实验将Finesse与传统的N-transform SF进行对比实验，测量在不同数据集上的差量压缩率（DCR）和差量压缩效率（DCE）。DCR反映了差量压缩的空间缩减效果，DCE反映了相似块之间的相似程度。实验结果如图6所示。

|  |  |
| --- | --- |
| C:\Users\ASUS\AppData\Roaming\Tencent\Users\1332765311\TIM\WinTemp\RichOle\`NTKY{FU%CVV9I%][YK$RCX.png | C:\Users\ASUS\AppData\Roaming\Tencent\Users\1332765311\TIM\WinTemp\RichOle\HL(OJP237SXCK}_DWOBP81D.png |

**图6 实验对比结果**

实验结果可知，Finesse与N-transform SF具有相近的压缩率，而Finesse可以检测到更加相似的数据块。在相似度计算阶段，Finesse在两台机器上分别实现了3.5x和3.2x的速度提升，大大改善了系统的I/O性能。

### 3.4 论文总结

论文针对当前的差量压缩相似性检测部分的主流算法N-transform SF进行研究，提出了Finesse的算法。根据高度相似的块的细粒度特征局部性，算法将数据块划分为多个子块并从每个子块中提取特征来，从而减少了相似度检测的计算开销，极大提升了I/O性能。

## 4 基于数据恢复的优化

论文发表于FAST 18，论文名称：ALACC: Accelerating Restore Performance of Data Deduplication Systems Using Adaptive Look-Ahead Window Assisted Chunk Caching[9]

### 4.1 论文背景

重复数据删除技术已广泛应用于存储系统中以提高空间利用率。恢复原始数据是重复数据删除的反向过程。在重复数据删除系统中，由于数据块分散在许多容器中，数据恢复时会频繁读取容器，因此数据块碎片会严重阻碍数据恢复性能，造成I/O性能的下降。

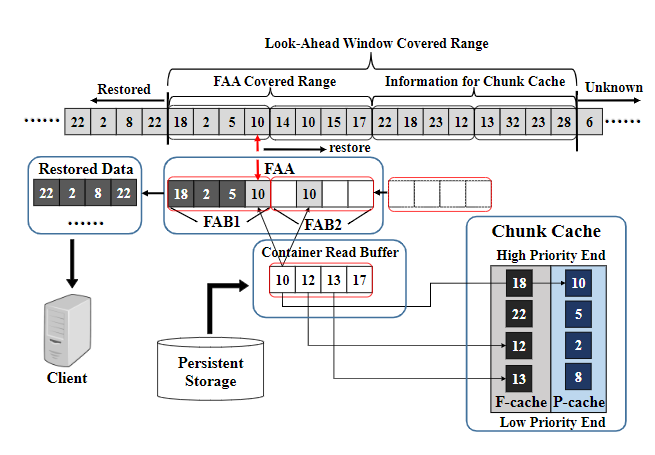
目前基于容器的缓存，基于块的缓存和前向组装等的几种方案是当前主流的用于减少恢复过程中的磁盘读取次数的方法。基于容器的缓存具有较低的运行成本，但是基于块的缓存可以更好地过滤出与将来恢复无关的数据块，从而更好地提高缓存空间的利用率。Lillibridgeet等人提出的一种基于块的缓存的特殊方式称为前向组装[16]。该方法将多个容器用作称为前向装配区（FAA）的组装缓冲区，并且使用大小相同的前瞻窗口来标识要在下一个容器中恢复的数据块。与基于块的缓存相比，前向汇编的开销较低。如果每个唯一的数据块将在恢复后的短时间内重新出现，则前向汇编会非常有效

### 4.2 方法实现

论文首先通过不同数据集对比分析基于容器的缓存、基于块的缓存和前向组装的缓存效率。通过对比分析可知以下几点：①在大多数情况下，缓存的容器中的某些数据块与当前和不久的将来的恢复过程无关，同时有用的数据块可能会被迫与整个容器一起被逐出缓存。因此，缓存块比缓存容器效果更好。②前向组装的计算时间比基于块的缓存的计算时间低得多。如果大部分数据块时非重复数据块或者数据块重用的距离小于FAA的范围，则前向组装的性能会优于基于块的缓存。否则，它的性能会优于前向组装。

由于前向组装和基于块的缓存在不同场景下各有优势，所以论文希望将二者结合起来，利用前瞻窗口(LAW)、FAA和块缓存来进行数据恢复。LAW分成两个部分，第一部分用于前向组装，第二部分应用于基于块的缓存。其中F块表示根据LAW得知的未来会用到的数据块，P块表示FAA刚刚组装完的数据块。作者没有固定FAA，LAW和块缓存的大小，而是设计了一个动态自适应算法ALACC来适应不同的工作负载。ALACC可以动态调整FAA和块缓存的存储空间比例以及LAW的大小，如图7所示。算法首先检测调整FAA的大小的条件。如果条件满足，FAA将相应地增加1个容器大小，而块缓存的大小将相应地减少1个容器。否则，我们将检查调整块缓存大小的条件。最后，算法会修改LAW的大小。

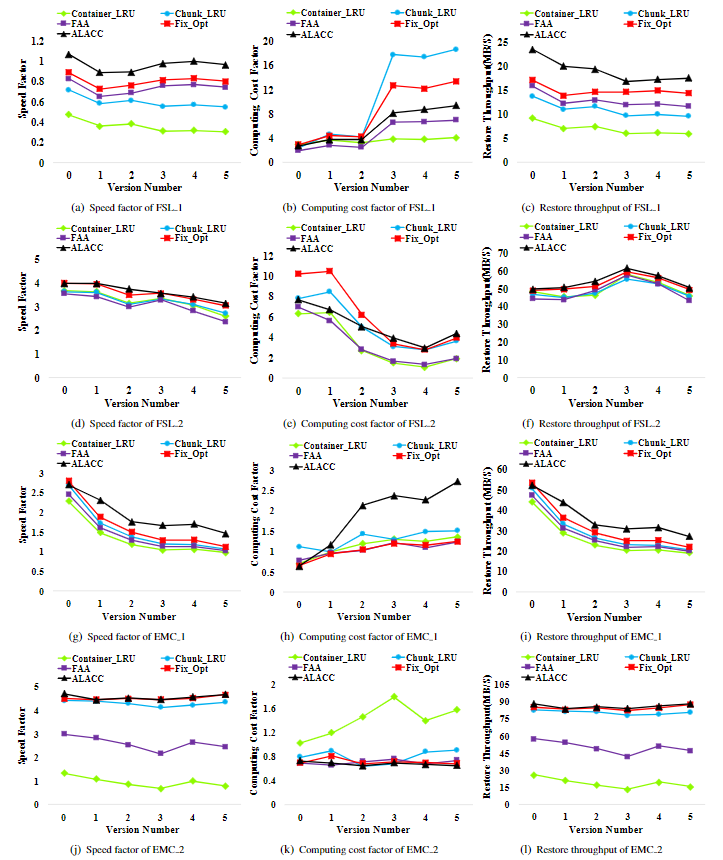
具体调整策略：首先，对于FAA来说，如果连续的FAA有效组装周期数大于给定阈值或者在组装周期中数据块中80％以上的重用距离小于当前FAA大小增加1个容器，则将FAA大小增加1个容器。在将FAA大小增加1个容器之后，将连续的FAA有效组装周期的计数重置为0，LAW大小减少一个容器大小。其次，如果FAA的大小没有发生改变，那么考虑调整块缓存的大小。如果块缓存中P块的数目变为0或者新增的F块的数量大于一个容器大小，块缓存增加一个容器的大小，同时LAW大小减小一个容器。如果P块的数量非常大，则块缓存大小将减少一个容器。此时，LAW的大小根据以下两个条件进行调整：①FAB中很少有数据块可在将来重用；②LAW的大小太小，无法为寻找足够多的F块。对于条件1，LAW大小减小一个容器。对于条件2，LAW大小增加K个容器大小，K等于当前LAW大小与最大LAW大小的差值除以FAA和块缓存的大小之和。最后，LAW的大小也可以单独调整。如果F块数量小于给定的阈值，LAW增加一个容器大小。如果高于某个阈值，LAW减小一个容器大小。



**图7 ALACC算法**

### 4.3 实验评估

为了评估ALACC的性能，论文采用了五个恢复算法进行对比，分别是ALACC，基于LRU的容器缓存，基于LRU的块缓存，FAA和FAA与基于块缓存的固定组合Fix\_Opt。在四个数据集上的测试表现如图8所示。



**图8 实验测试结果**

实验结果图中从左到右分别表示计算速度，计算开销和吞吐量。对于所有4个数据集，ALACC的总体平均速度最快，吞吐量最高。ALACC至少可以达到与Fix\_Opt相似或更好的性能。

### 4.4 论文总结

论文研究了不同数据恢复机制的性能，设计了一种称为ALACC的自适应算法。算法能够根据数据块的变化来自适应地调整FAA，块缓存和LAW的大小。ALACC的数据恢复性能要比基于容器的缓存，基于块的缓存和前向组装更好。

## 5 总结

由于数据中心存在大量冗余数据，影响了存储的性能。作为消除冗余数据，提高存储空间和网络传输带宽的关键技术，重复数据删除技术已经成为数据中心中不可缺少一部分。如何降低重复数据删除过程中的性能开销，提高数据吞吐量是重复数据删除更好地应用于数据中心的关键之一。以上三篇论文分别从指纹索引，差量压缩和数据恢复三个方面对重复数据删除技术进行了优化，提高了I/O性能。

参考文献

1. L. DuBois, M. Amaldas, and E. Sheppard, “Key considerations as deduplication evolves into primary storage,” White Paper 223310, 2011
2. A. El-Shimi et al., “Primary data deduplication - large scale study and system design,” USENIX Annual Technical Conference, 2012
3. D. T. Meyer and W. J. Bolosky, “A study of practical deduplication,” ACM Transactions on Storage (TOS), 2012.
4. P. Shilane et al., “WAN optimized replication of backup datasets using stream-informed delta compression,” USENIX Annual Technical Conference, 2012
5. L. DuBois, M. Amaldas, and E. Sheppard, “Key considerations as deduplication evolves into primary storage,” White Paper 223310, Mar. 2011. [Online]. Available: <http://www.bedrock-tech.com/wp-content/uploads/2010/05/wp_key-considerations.pdf>
6. W. Xia et al., “A comprehensive study of the past, present, and future of data deduplication,” Proceedings of the IEEE, vol. 194, no. 9, pp. 1681-1710, 2016
7. G. Xu, B. Tang, H. Lu, Q. Yu and C. W. Sung, "LIPA: A Learning-based Indexing and Prefetching Approach for Data Deduplication," 2019 35th Symposium on Mass Storage Systems and Technologies (MSST), Santa Clara, CA, USA, 2019, pp. 299-310, doi: 10.1109/MSST.2019.00010.
8. Zhang Y, Xia W, Feng D, et al. Finesse: Fine-grained feature locality based fast resemblance detection for post-deduplication delta compression[C]//17th {USENIX} Conference on File and Storage Technologies ({FAST} 19). 2019: 121-128.
9. Cao Z, Wen H, Wu F, et al. {ALACC}: Accelerating restore performance of data deduplication systems using adaptive look-ahead window assisted chunk caching[C]//16th {USENIX} Conference on File and Storage Technologies ({FAST} 18). 2018: 309-324.
10. B. Zhu, K. Li, and R. H. Patterson, ”Avoiding the Disk Bottleneck in the Data Domain Deduplication File System,” in Proc. USENIX FAST, 2008.
11. M. Lillibridge, K. Eshghi, D. Bhagwat et al., ”Sparse indexing: Large scale, inline deduplication using sampling and locality.” in Proc. USENIX FAST, 2009.
12. D. Bhagwat, K. Eshghi, D. D. Long et al., ”Extreme binning: Scalable, parallel deduplication for chunk-based file backup,” in Proc. IEEE MASCOTS, 2009
13. L. Zhou, ”A survey on contextual multi-armed bandits”, arXiv preprint arXiv:1508.03326
14. M. Fu, D. Feng, Y. Hua, X. He, Z. Chen, W. Xia, Y. Zhang, and Y. Tan, “Design tradeoffs for data deduplication performance in backup workloads,” in Proc. USENIX Conference on File and Storage Technologies (FAST’15). Santa Clara, CA, USA, 2015, pp.331-344
15. SHILANE, P., HUANG, M., WALLACE, G., AND ET AL. WAN optimized replication of backup datasets using stream-informed delta compression. In the 10th USENIX Conference on File and Storage Technologies (FAST’12) (San Jose, CA, USA, 2012), USENIX Association, pp. 49–63
16. Mark Lillibridge, Kave Eshghi, and Deepavali Bhagwat. Improving restore speed for backup systems that use inline chunk-based deduplication. In 11th USENIX Conference on File and Storage Technologies (FAST 13), pages 183–198, 2013