R-dedup:一种重复数据删除指纹计算的优化方法

王龙翔[1]，董凯[2]，王鹏博[1]，董小社[1]，张兴军[1]，朱正东[1]，张利平[1]
(1.西安交通大学计算机科学与技术学院，710049，西安；2.西安美术学院信息中心，710065，西安)

摘要：为减缓存储系统中传统重复数据删除方法在高性能固态存储盘中存在的指纹计算性能瓶颈，提出了重复数据删除指纹计算的性能优化方法R-dedup。在基于内容分块算法基础上，将切分后形成的所有数据块进一步切分为更小粒度的48 B等长数据片。基于Rabin哈希长度小于原始数据、多个Rabin哈希同时发生碰撞概率极低、数据片的Rabin哈希可以重复利用基于内容分块算法在滑动窗口过程中产生的计算结果的基础，利用数据片的Rabin哈希替代原始数据，并将其作为数据块的SHA-1指纹输入，减少SHA-1函数数据计算量，提高指纹计算性能。选取Linux内核、Imagenet等5组具有代表性的数据集，对R-dedup和标准基于内容分块的重复数据删除方法在数据分块性能、指纹计算性能、索引表检索性能和I/O性能方面分别进行了比较。结果表明：R-dedup的数据分块性能、索引表检索性能、I/O性能与对比方法具有4%左右的误差波动，性能基本一致；R-dedup的指纹计算吞吐率是对比方法的165%～422%，总体吞吐率是对比方法的6%～54%。

关键词：存储系统；重复数据删除；固态存储盘；Rabin哈希；性能优化
中图分类号：TP319 **文献标志码**：A
DOI：10.7652/xjtuxb202101006 **文章编号**：0253-987X(2021)01-0043-09

OSID码

R-dedup: A Performance Improvement Strategy for Fingerprint Calculation of Data De-Duplication

WANG Longxiang[1], DONG Kai[2], WANG Pengbo[1], DONG Xiaoshe[1],
ZHANG Xingjun[1], ZHU Zhengdong[1], ZHANG Liping[1]
(1. School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China;
2. Information Center, Xi'an Academy of Fine Arts, Xi'an 710065, China)

Abstract: To alleviate the performance bottleneck of SHA-1 fingerprint computing of data de-duplication in high-performance solid-state storage system, we propose an improvement strategy for fingerprint calculation of data de-duplication, R-dedup, which uses Rabin hash instead of original data to reduce SHA-1 fingerprint calculation. Based on content defined chunking algorithm, R-dedup further divides each data chunk formed after segmentation into smaller 48-byte length data slices. Owing to the shorter length of Rabin hash than the original data slice, the collision probability of multiple Rabin hashes is very low, and the Rabin hash of data slice can be obtained from the intermediate calculation results of sliding window of content defined chunking algorithm. The Rabin hash of data slice is used to replace the original data as the SHA-1 fingerprint input value of the data chunk, so as to reduce the data calculation amount of SHA-1 function and improve the performance. Five representative datasets, including Linux kernel and

收稿日期：2020-06-19。 作者简介：王龙翔(1988—)，男，工程师；董小社(通信作者)，男，教授，博士生导师。 基金项目：国家重点研发计划资助项目(2018YFB0203902)。
网络出版时间：2020-08-25 网络出版地址：http://kns.cnki.net/kcms/detail/61.1069.T.20200825.1348.002.html

imagenet, are selected to compare the data chunking performance, SHA-1 computing performance, index table retrieval performance and I/O performance between R-dedup and standard content defined chunking of data de-duplication. The results show that R-dedup has only about 4% error fluctuation compared with the standard content defined chunking method in terms of data chunking performance, I/O performance and index retrieval performance. Compared with the standard content defined chunking strategy, R-dedup can improve the performance of SHA-1 fingerprint computing by 165%-422% and the overall system throughput by 6%-54%.

Keywords: storage system; data de-duplication; solid-state drive; Rabin hash; performance improvement

近年来,随着半导体技术的不断发展,固态硬盘(SSD)的每字节存储成本在不断下降。无论是个人计算机还是数据中心服务器,SSD 正在逐渐取代机械硬盘(HDD)成为主要存储设备。与 HDD 相比,SSD 具有更高的性能和更低的功耗,并且更加抗振动干扰。随着 3DXpoint 等新型非易失性介质(NVM)的发展,基于 NVM 的新一代 SSD 将具有更加优异的性能。

重复数据删除作为一项高效的数据精简技术,在存储系统中得到了广泛应用[1]。随着 SSD 成为了主流的存储设备,如何针对 SSD 的特点优化重复数据删除方法的性能成为了研究者关注的问题[2-5]。Gupta 等指出,在普通文件系统的 I/O 负载中存在重复数据,对 SSD 写入数据进行重复数据删除是有效的[6]。Chen 等提出 CaFTL 方法,在 SSD 内部 FTL 层实现了重复数据删除[7]。通过在 SSD 的写入路径中识别重复数据,避免这些数据写入 flash 介质中,可以延长 SSD 寿命。重复数据删除需要对数据块计算 SHA-1 哈希值作为指纹进行数据查重操作。然而,计算 SHA-1 哈希值是一项计算密集型任务,在 SSD 的写入路径中加入重复数据删除会增加写入延迟,降低数据写入吞吐率。在基于 NVM 的 SSD 中,由于数据 I/O 操作延迟不断降低,吞吐率不断增加,该问题会更加严重。文献[8]指出,在传统 HDD 设备中,基于 SHA-1 的指纹计算执行时间约占重复数据删除操作总时间的 21%,在 NVM 存储系统中,这一比例上升至 44%。因此,如何降低 SHA-1 指纹计算开销成为了提高高性能 SSD 中重复数据删除性能的关键问题。

为了降低 SHA-1 指纹计算时间开销,CaFTL 提出了采样和预哈希方法[7]。该方法假设上层负载中只存在少量重复数据,因此每次请求都会被抽样,只有当某次写入请求中被抽样数据块为重复时才进行重复数据删除操作,否则认为该次请求包含重复数据的概率较低,不对该次请求进行重复数据删除。同样,Wang 等提出了类似的思想,在执行重复数据删除之前,对数据集进行采样,以获得估计的重复数据删除率[8]。只有重复率高的数据集才会执行重复数据删除,而重复率低的数据集则不会执行。采样方法通过减少需要进行重复数据删除操作的数据量来降低指纹计算开销,存在的问题是会降低重复数据删除率。Chen 等提出的 NF-Dedup 用弱哈希算法(如 CRC)替代 SHA-1 作为数据块指纹,以此来减少指纹计算开销[9]。该方法的原理是 CRC 等弱哈希算法的计算开销远小于 SHA-1 这类强哈希函数的。当弱哈希在指纹库中检索不到时,可以确定对应的数据块不存在,但是当检索到弱哈希存在时,不能确定对应的数据块是否存在。这是由于弱哈希具有较高的哈希碰撞率,即若干个不同的数据块可能具有相同的弱哈希值。因此,NF-Dedup 通过逐字节比较所有具有相同 CRC 指纹的数据块来判断是否为重复数据块。该方法随着存储数据量的增大,产生哈希碰撞的几率会持续上升。例如:在存储数据量较小时,可能存在两个数据块具有相同的 CRC 哈希值,当存储数据量增大后,具有相同 CRC 哈希值的数据块可能上升为 10。此时,需要对 10 个数据块都进行逐字节对比才能判断数据块是否重复,这会显著增加数据写入延迟。

为了解决上述问题,本文在弱哈希方法思想基础上,提出了面向基于内容分块算法(CDC)的重复数据删除指纹计算的优化方法 R-dedup。CDC 能够实现可变长度分块,是目前应用最广泛的重复数据删除分块算法[10-11]。在此算法基础上,又衍生出了许多改进算法,如 TTTD[12]、Bimodal CDC[13]、fastCDC[14]等。由于这些算法都是基于 CDC 进行的改进,因此 R-dedup 也可应用于这些算法。R-dedup 的核心思想是将数据块划分为更小粒度的 48 B 等长数据片,利用 CDC 滑动窗口过程中已计算的

数据片 Rabin 指纹，替代原始数据作为 SHA-1 函数的输入，计算数据块对应的 SHA-1 指纹。因为 Rabin 指纹长度通常为 10 B，远小于 48 B，因此 R-dedup 减少了 SHA-1 函数的输入数据量。Rabin 指纹可以直接利用 CDC 算法滑动窗口过程中生成的结果，无需重复计算。实验结果表明，R-dedup 的 SHA-1 指纹计算吞吐率提升至标准 CDC 方法的 165%～422%，系统总体吞吐率提升至标准 CDC 方法的 6%～54%。

1　背景知识

1.1　基于内容的分块算法

重复数据删除需要先对数据流进行切分，将其划分为更细粒度的数据块，以数据块为单位进行重复数据检测。可变长度分块能够避免因为字节变化造成的数据块漂移问题，检测到更多的重复数据，因此目前被重复数据删除系统广泛使用。

CDC 是目前可变长度分块的标准算法。CDC 的主要原理如图 1 所示，通过设置大小为 48 B 的滑动窗口，从数据流的起始处开始滑动，计算窗口内数据的 Rabin 指纹[15]。之后，通过将 Rabin 指纹和掩码进行与操作，判断所得结果是否等于预先设定的魔数。如果等于预先设定的魔数，则将当前窗口的最后一个字节作为分块的边界点，当窗口在数据流中滑动结束后，所形成的边界点将数据流划分为长度不同的一系列数据块集合。为了避免出现分块过大或过小的情况，可设置分块大小的最小值与最大值。当分块大小小于最小值时，舍弃当前边界点；当分块大小大于最大值时，强制设置当前窗口最后一个字节为边界点。

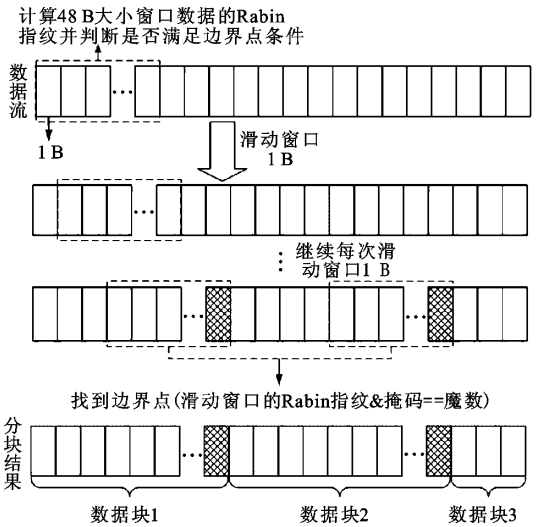


图 1　CDC 原理

1.2　Rabin 指纹计算

Rabin 指纹由哈佛大学的 Rabin 于 1981 年提出。对于任何一个 n 位字符串 $b_0,\cdots,b_{n-1}$，可将其表示为有限域 GF(2)上的 $n-1$ 次多项式

$$f(x)=b_1x^{l-1}+b_2x^{l-2}+\cdots+b_l \tag{1}$$

随机选择有限域 GF(2)上的 k 次不可约多项式 $p(x)$，定义字符串 m 的 Rabin 指纹为 GF(2)上 $f(x)$除以 $p(x)$后的余数，即 $k-1$ 次多项式 $r(x)$

$$r(b_1,\cdots,b_l)=(b_1x^{l-1}+b_2x^{l-2}+\cdots+b_l)\bmod p(x)=r_1x^{k-1}+r_2x^{k-2}+\cdots+r_k \tag{2}$$

在 CDC 算法中，设置一个大小为 48 B 的数据窗口，计算当前窗口的 Rabin 指纹，判断是否为块边界。滑动 1 B 后，窗口数据发生变化，因此需要重新计算当前窗口的 Rabin 指纹。如果再次用式(2)计算 Rabin 指纹，会造成不必要的计算开销。事实上，当前窗口与滑动前窗口的数据只有一个字节不一样，因此可以利用上一个窗口 Rabin 指纹来计算当前窗口 Rabin 指纹，公式为

$$\begin{aligned}f(b_9,\cdots,b_{l+8})&=(b_9x^{l-1}+\cdots+b_{l+8})\bmod p(x)=\\&(x^8(b_9x^{l-9}+\cdots+b_l)+b_{l+1}x^7+\cdots+b_{l+8})\bmod p(x)=\\&(x^8(b_1x^{l-1}+\cdots+b_8x^{l-8}+b_9x^{l-9}+\cdots+b_l+b_1x^{l-1}+\cdots+b_8x^{l-8})+b_{l+1}x^7+\cdots+b_{l+8})\bmod p(x)=\\&(x^8(r_1x^{k-1}+r_2x^{k-2}+\cdots+r_k+b_1x^{l-1}+\cdots+b_8x^{l-8})\bmod p(x)+b_{l+1}x^7+\cdots b_{l+8})\bmod p(x)=\\&((x^8(f(b_1,\cdots,b_l)+(b_1x^{l-1}+\cdots+b_8x^{l-8})\bmod p(x))\bmod p(x)+(b_{l+1}x^7+\cdots+b_{l+8}))\bmod p(x)\end{aligned} \tag{3}$$

通过式(3)可以看出，通过对多项式做变化，可将 $f(b_9,\cdots,b_{l+8})$表示为 $f(b_1,\cdots,b_l)$的多项式，从而利用已计算窗口的 Rabin 指纹结果。与式(2)相比，式(3)只需要将 $f(b_1,\cdots,b_l)$做移位运算。

在窗口滑动 1 B(8 b)后，当前窗口从 $b_1,\cdots,b_l$ 变为 $b_9,\cdots,b_{l+8}$，而 $b_9,\cdots,b_{l+8}$的 Rabin 指纹可根据式(3)利用 $b_1,\cdots,b_l$ 已计算的 Rabin 指纹 $f(b_1,\cdots,b_l)$得到。

2　R-dedup 设计

R-dedup 的核心思想是将数据块划分为更小粒度的 48 B 等长小数据片，先计算每个数据片的 Rabin 指纹，再将数据块包含的所有数据片对应的 Rabin 指纹替代原始数据，用于计算数据块对应的 SHA-1 指纹。R-dedup 框架如图 2 所示。

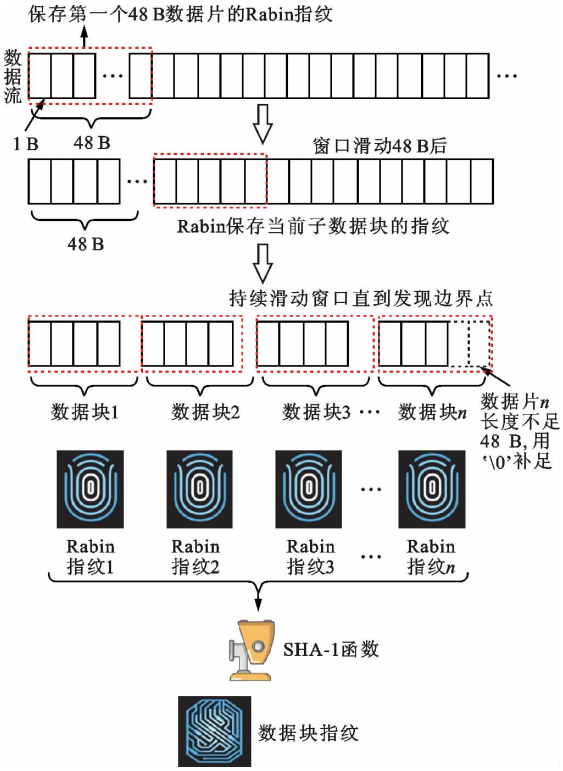


图 2 R-dedup 框架

2.1 重复数据检测

重复数据删除系统检测重复数据的原理是使用一个函数 $h(x)$，如果两个数据块 $a \neq b$，则 $h(a) \neq h(b)$，反之如果 $a=b$，则 $h(a)=h(b)$。SHA-1 哈希函数符合这一要求，目前普遍被重复数据删除系统作为 $h(x)$ 函数使用。R-dedup 通过计算 $f(g(x))$ 作为数据块的指纹来加速指纹计算，令 $g(x)=w(x_1)\cdot w(x_2)\cdot\cdots\cdot w(x_n)$，其中，$\cdot$ 是字符串连接运算符，$x_1,x_2,\cdots,x_n$ 是数据块 x 按 48 B 切分后形成的数据片，$w(x)$ 为弱哈希函数，例如 Rabin 哈希函数。由于 Rabin 哈希值的长度通常小于 80 b (10 B)，小于数据片 x_i 的长度 48 B，因此 $g(x)$ 的长度小于 x 的，$h(g(x))$ 的计算量小于 $h(x)$ 的。CDC 分块过程中需要不断计算 48 B 窗口的 Rabin 哈希值，因此只需每隔 48 B 保存一次 Rabin 哈希值，即可得到 $w(x_i)$，无需再次计算。R-dedup 算法伪代码如下。

输入：数据流

输出：SHA-1 指纹

```
1  获取数据流长度
2  计算数据片数量
3  初始化数据流指针为 0
4  初始化 48 B 窗口
5  初始化 Rabin 指纹数组
6  从数据流中读取 48 B 数据填满窗口，并将数据流指针加 48
7  while 数据流指针未指向数据流末尾 do
8     计算窗口数据的 Rabin 指纹
9     if Rabin 指纹形成边界点 then
10       跳出循环
11    end if
12    if 已滑动长度等于 48 B 的倍数 then
13    保存当前窗口数据的 Rabin 指纹值到数组
14    end if
15    数据流指针加 1
16 end while
17 if 窗口数据不足 48 B then
18    用'\0'填充窗口
19    计算窗口 Rabin 指纹
20    保存窗口数据的 Rabin 指纹值到数组
21 end if
22 将 Rabin 指纹数组作为输入，计算对应 SHA-1 指纹并返回
```

2.2 指纹计算

在 R-dedup 中，将 CDC 切分后形成的数据块再进一步切分成大小为 48 B 更细粒度的数据片。在 CDC 切分数据块的过程中，需要保存下所有 48 B 数据片对应的 Rabin 指纹。当数据块边界点寻找到后，用 48 B 数据片对应的 Rabin 指纹而不是原始数据流作为输入来计算数据块的 SHA-1 指纹。因此，R-dedup 可以减少 SHA-1 函数需要计算的数据量。例如，选择度为 64 的不可约多项式计算 Rabin 指纹，则 Rabin 指纹长度小于 64 b(8 B)。在用数据片 Rabin 指纹而不是 48 B 的原始数据做 SHA-1 计算后，R-dedup 可将 SHA-1 的输入数据大小至少减少到原始输入数据大小的 $8/48=1/6$。理论上，R-dedup 可将 SHA-1 的计算速度提高 6 倍以上。

2.3 数据块指纹碰撞

数据块发生指纹冲突是指两个不同的数据块具有相同的 SHA-1 哈希值。SHA-1 哈希函数本身就可能发生哈希碰撞，但是由于这一概率极低，所以目前学术界普遍认为 SHA-1 哈希碰撞问题可以忽略。Venti 系统是最早采用重复数据删除的存储系统，该系统采用了 SHA-1 函数生成数据块指纹。文献[16]指出：在 EB 级存储系统中，发生指纹哈希碰撞的概率为 10^{-20}，碰撞概率足够小，可以忽略不计。因此，只要哈希碰撞的概率足够低，在工程应用中就可以忽略哈希碰撞问题。R-dedup 通过 Rabin 指纹

替代原始数据进行 SHA-1 指纹计算，如果 Rabin 指纹发生碰撞，则数据块对应的 SHA-1 指纹也会发生碰撞。

对于两个不同数据块 a 和 b，R-dedup 会将其划分为更细粒度的 48 B 数据片。假设数据片分别为 $\langle a_1,\cdots,a_n\rangle$，$\langle b_1,\cdots,b_n\rangle$，所有的数据片发生 Rabin 指纹碰撞，则 a 和 b 就会被错误判断为相同数据块。文献[17]指出，两个数据块对应的 Rabin 指纹发生碰撞的概率小于 $n_0^2m/2^k$，其中，n_0 为可能发生冲突的数据块数，m 为数据块的位长度，k 为不可约多项式的次数。若 a 和 b 由 i 个数据块组成，则 a 和 b 被错误判断为相同数据块的概率为 $(n_0^2m/2^k)^i$。

为了避免指纹冲突，可以设置更大的预期块长度。较长的数据块有更多的 48 B 数据片，所有数据片同时发生 Rabin 哈希碰撞的概率将更低。此外，还可以使用度更高的不可约多项式来计算 Rabin 哈希，以降低 Rabin 哈希碰撞的概率。

考虑一个比较极端的大规模存储场景。假设存储系统保存了 1 EB 的非重复数据，如果数据重复率为 10∶1，则实际存储了 10 EB 数据。由此可得，48 B 数据块数 $n_0=2^{60}/48$，$m=48\times8$，碰撞概率为

$$\frac{(2^{60}\times48^{-1})^2\times48\times8}{2^k}=\frac{2^{119-k}}{3}\tag{4}$$

如果选择一个度为 200 的不可约多项式，则发生 Rabin 指纹碰撞的概率为 $2^{-41}/3\approx1.38\times10^{-25}$。因此，如果想降低 Rabin 指纹碰撞概率，可根据数据存储规模选择度更高的不可约多项式。

3　实　验

3.1　实验环境

本文实验均在同一台高性能服务器下进行，服务器具体配置如下：① CPU，2 路英特尔 Xeon E5-2650 v2 @ 2.6 GHz 处理器；② 内存，128 G DDR4 内存；③操作系统，Ubuntu 16.04；④存储设备，Memblaze Pblaze3 PCIe 1TB SSD。

重复数据删除参数设置如下：①预期分块大小平均值，4 096、8 192、16 384、32 768、65 536 B；②预期分块大小最大值，131 072 B；③预期分块大小最小值，1 024 B；④不可约多项式，0xbfe6b8a5bf378d83。

3.2　数据集

收集了 5 组具有代表性的数据集进行测试。为了保证 R-dedup 结果可以复现，所选的 5 组数据集都是公开数据集，可通过互联网进行下载。

5 组数据集情况如下：①Linux 内核源代码[18]，从版本 4.01 到 4.19，包含 641 个版本的内核源码数据，该数据集记为 kernel；②著名机器学习网站 kaggle 上的视频数据集[19]，包含大量用于机器学习的视频文件，该数据集记为 video；③Flickr30k 数据集[20]，用于机器学习的基准测试数据集，包含大量来自 Flickr 网站的图像文件，该数据集记为 image1；④斯坦福计算机视觉视频数据集[21]，由斯坦福大学研究团队录制的斯坦福校园场景视频数据集，用于计算机视觉研究，该数据集记为 video2；⑤Imagenet 数据集[22]，用于机器学习的著名公开图像训练数据集，包含大量图像数据，每一类图像数据存储为一个 tar 包，该数据集记为 image2。

3.3　数据集重复率信息

预期分块大小是重复数据删除系统在分块过程中期望形成数据块的大小，但是受到文件自身大小等因素的限制，实际分块大小可能与预期分块大小差异较大。在预期分块大小为 4、8、16、32、64 kB 的情况下，分别对 R-dedup 方法和标准 CDC 方法进行了实验，结果如表 1 所示，表中重复数据率为原始数据大小与重复数据删除后实际存储数据大小的比值，反映了数据冗余程度。可以看出：kernel 数据集由于包含数量众多的不同版本 Linux 内核源文件，存在大量冗余数据，因此具有相当高的重复数据率，达到了 70.13～134.67，其他两组数据集的重复数据率则在 2 左右。另外，由于 kernel 数据集由大量小文件组成，而实际分块大小必定小于文件本身大小。因此，在设置预期分块大小大于 8 kB 以后，实际分块大小也不会明显上升。

3.4　数据分块吞吐率对比

数据分块吞吐率对比结果如图 3 所示。可以看出，R-dedup 方法与标准 CDC 方法没有显著的数据分块吞吐率差异。这是由于 R-dedup 与标准 CDC 方法在分块过程中的唯一差别是 R-dedup 会临时保存数据片的 Rabin 指纹，这一操作并不会对性能造

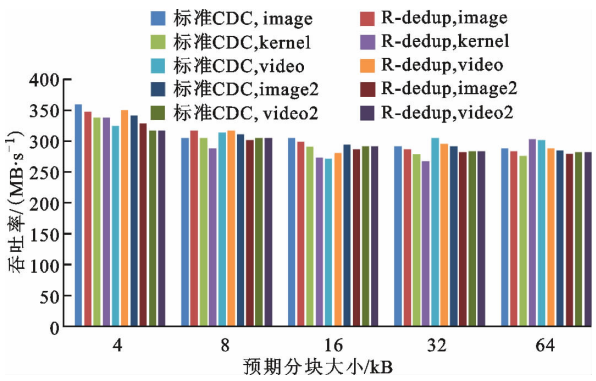


图 3　数据分块吞吐率对比

表 1 重复数据删除实验结果

数据集	重复数据删除率	预期分块大小/kB	实际分块大小/B	分块数量	实际存储数据量/GB
image1	2.01	4	4 961	1 785 624	4.11
	2.01	8	8 583	1 032 196	4.11
	2.00	16	15 471	572 640	4.12
	2.00	32	25 138	352 422	4.12
	2.00	64	35 207	251 636	4.12
kernel	134.67	4	3 705	116 437 822	2.98
	108.31	8	8 583	78 593 131	3.71
	89.52	16	7 303	59 067 142	4.49
	77.37	32	8 821	48 906 792	5.19
	70.13	64	9 848	43 803 692	5.73
video1	2.03	4	5 115	761 842	1.78
	2.03	8	9 159	425 516	1.78
	2.03	16	16 819	231 714	1.78
	2.03	32	28 430	137 086	1.78
	2.03	64	40 730	95 688	1.78
image2	1.01	4	5 141	28 765 488	137.01
	1.00	8	9 185	16 101 626	137.07
	1.00	16	17 250	8 573 657	137.19
	1.00	32	29 357	5 037 795	137.30
	1.00	64	42 227	3 502 369	137.43
video2	1.00	4	5 135	14 106 014	67.32
	1.00	8	9 219	7 839 991	67.32
	1.00	16	17 096	4 227 783	67.32
	1.00	32	29 211	2 474 341	67.32
	1.00	64	42 043	1 719 151	67.32

成显著影响。暂存 Rabin 指纹的内存空间在当前数据块的 SHA-1 指纹计算完成后即可释放，仅造成少量的内存空间开销。例如，如果预期分块大小为 4 kB，则将 4 kB 分块再切分为 48 B 的数据片后，共形成 4×1 024/48≈86 个数据片。假设 Rabin 指纹长度为 10 B，则需要临时分配 860 B 的内存空间。

3.5 SHA-1 指纹计算吞吐率对比

SHA-1 指纹计算吞吐率对比结果如图 4 所示。可以看出，R-dedup 的总体吞吐率提升至标准 CDC 方法的 165%～422%。R-dedup 方法有效降低了 SHA-1 指纹的计算时间开销，提升了吞吐率。这是由于 R-dedup 利用 Rabin 指纹减少了 SHA-1 函数需要计算的数据量，从而降低了计算开销。

3.6 数据读写吞吐率对比

数据读写吞吐率对比结果如图 5 和图 6 所示。可以看出，R-dedup 和标准 CDC 没有显著的数据读写吞吐率差异。这是因为 R-dedup 和标准 CDC 方法在数据读写方面没有区别，都是使用标准 POSIX 接口对数据进行读写操作，吞吐率主要受到数据集自身特点以及 SSD 本身性能的影响。kernel 数据集由大量小文件构成，因此在读取过程中需要进行

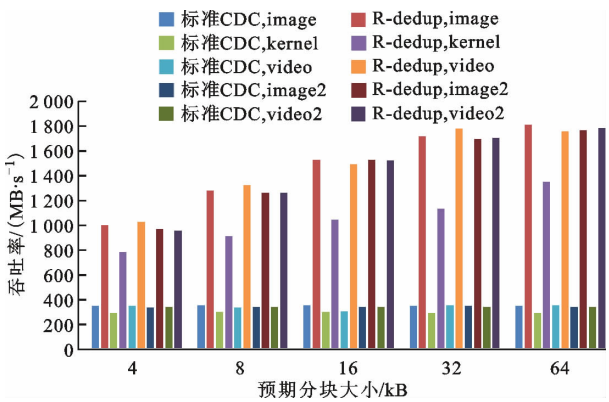


图 4 SHA-1 指纹计算吞吐率对比

大量文件打开、关闭操作，造成了大量性能开销，导致读取性能较低，吞吐率只有约 30 MB/s。其他 3 组数据集读取吞吐率可达 2 GB/s 以上，写入吞吐率可达 400 MB/s～1 GB/s。

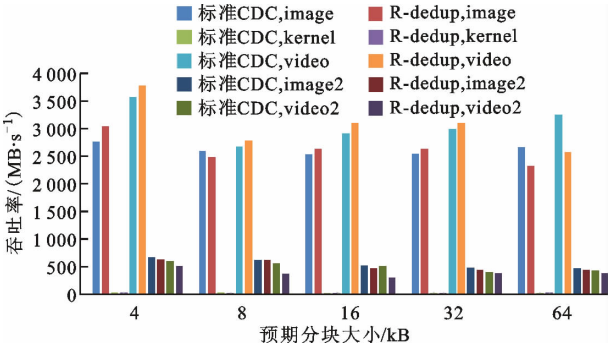


图 5　数据读取吞吐率对比

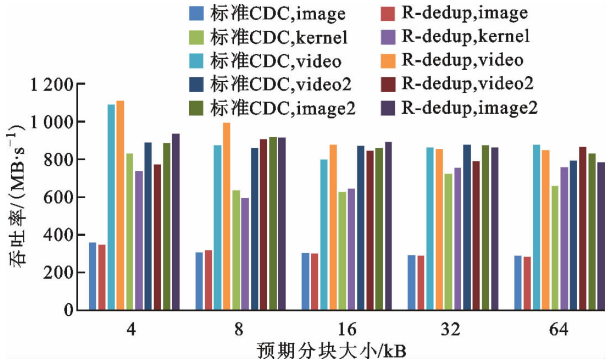


图 6　数据写入吞吐率对比

3.7　指纹查寻时间对比

指纹查询时间对比结果如图 7 所示。可以看出，R-dedup 和标准 CDC 方法的指纹查询时间基本相同。指纹索引表采用哈希表结构实现，同样存储在 SSD 中，同时在内存中维持缓存提升性能。指纹查询时间与指纹数量成正比，kernel 与 image2 数据集消耗的时间最多。预期分块数量设置越大，产生的指纹数越少，查询时间也会减少。由于 R-dedup 产生的指纹数和标准 CDC 方法的一样，因此两种方法的指纹查询时间基本相同。

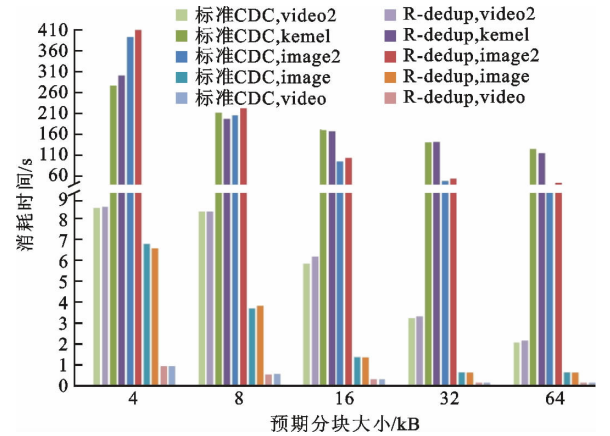


图 7　指纹查询时间对比

3.8　系统总体吞吐率对比

重复数据删除系统的总体吞吐率主要受分块性能、SHA-1 指纹计算性能以及数据块读写性能影响。由于 R-dedup 相比标准 CDC 方法有效提升了 SHA-1 指纹计算性能，且在分块性能、数据读写性能方面没有明显差异，所以 R-dedup 相比标准 CDC 方法提升了总体吞吐率。主要操作消耗时间分布对比如图 8 所示，可以看出，相比标准 CDC，R-dedup 方法能够提升总体性能约 6%～54%。性能提升程度取决于 I/O 等操作开销所占比例，例如，kernel 数据集中读操作占总体时间开销比例大，因此即使 R-dedup 显著降低了 SHA-1 指纹计算开销，提升的系统整体吞吐率也较为有限。

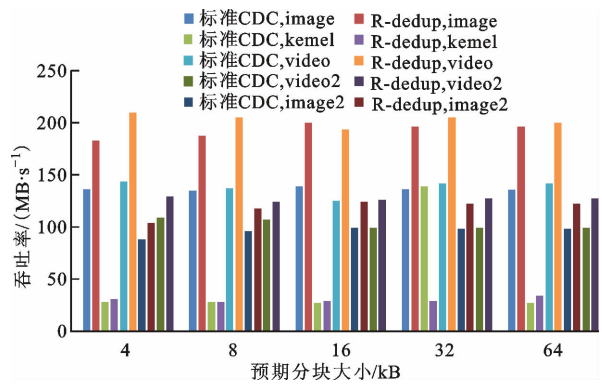
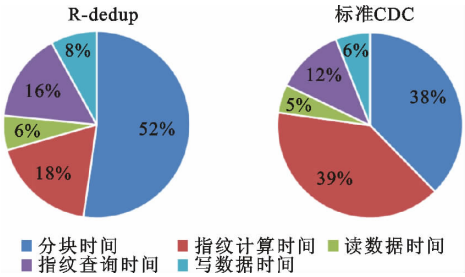


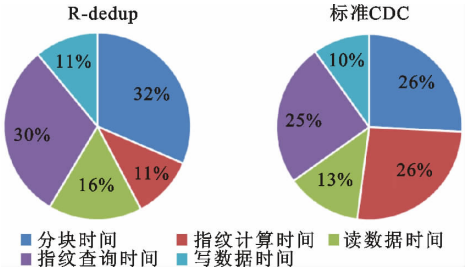
图 8　系统总体吐率对比

3.9　主要操作消耗时间分布对比

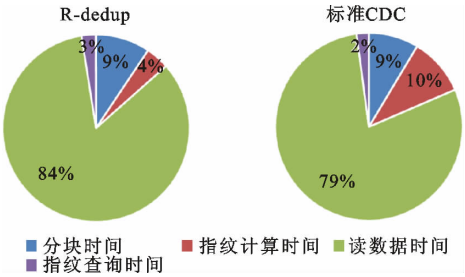
在预期分块大小设置为 4 096 B 时，R-dedup 和标准 CDC 方法主要操作时间消耗分布如图 9 所示。重复数据删除消耗时间由 I/O 时间、分块时间和指纹计算时间构成，I/O 时间由数据读写时间和指纹查询时间构成。随着数据规模的增长，指纹索引表过大无法完全维持在内存中。因此，目前主流重复数据删除系统采用内存作为缓存、外存作为存储的方式保存索引表。在指纹查询过程中，如果缓存未命中则会产生 I/O 操作。在标准 CDC 方法中，除了 kernel 数据集由于读时间占比过高，导致指纹计算比例较小(10%)外，其他 4 组数据集的指纹计算比例为 26%～39%，印证了在高性能 SSD 中，SHA-1 指纹计算成为了主要的性能瓶颈的现象。R-dedup 方法中指纹计算比例下降为 4%～20%，验证了 R-dedup 方法的有效性。



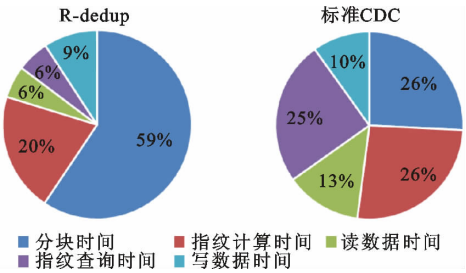
(a)image1 数据集



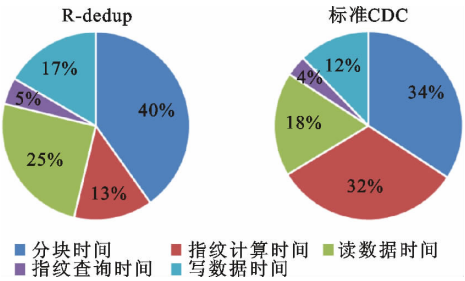
(b)image2 数据集



(c)kernel 数据集



(d)video1 数据集



(e)video2 数据集

图 9 主要操作消耗时间分布对比

4 结 论

重复数据删除作为一种高效的数据精简技术，在存储系统中得到了广泛应用，然而随着高性能 SSD 的不断普及，重复数据删除系统的主要性能瓶颈由 I/O 操作变为了 SHA-1 指纹计算。本文提出的 R-dedup 方法通过用 Rabin 指纹替换原始数据作为 SHA-1 指纹的输入来减少计算量，缓解了 SHA-1 指纹性能瓶颈。本文得出的主要结论如下。

(1)提出了利用 Rabin 哈希替代原始数据以减少 SHA-1 指纹计算的重复数据删除指纹计算优化方法 R-dedup。R-dedup 在基于内容分块算法基础上，将切分后形成的每一个数据块进一步切分为更小粒度的 48 B 等长数据片。基于 Rabin 哈希长度小于原始数据、多个 Rabin 哈希同时发生碰撞概率极低以及数据片的 Rabin 哈希可从基于内容分块算法滑动窗口的中间计算结果获取的观察，用数据片的 Rabin 哈希替代原始数据作为数据块的 SHA-1 指纹输入，以减少 SHA-1 函数数据计算量，提高指纹计算性能。

(2)相比标准 CDC，R-dedup 在 I/O 性能、索引表检索性能方面与标准基于内容分块的重复数据删除方法只有 4% 的误差波动，性能基本一致。R-dedup 提升 SHA-1 指纹计算吞吐率达到 165%～422%，提升重复数据删除系统总体性能达到 6%～54%。R-dedup 较好地缓解了高性能 SSD 中重复数据删除系统的 SHA-1 哈希计算性能瓶颈。

未来将进一步选取规模更大的数据集做更大范围测试，观察 R-dedup 是否会出现 SHA-1 哈希碰撞问题，并分析发生碰撞对实际存储系统可能造成的影响。

参考文献：

[1] 付印金，肖侬，刘芳，等. 基于重复数据删除的虚拟桌面存储优化技术 [J]. 计算机研究与发展，2012，49(Z1)：125-130.
FU Yinjin，XIAO Nong，LIU Fang，et al. Deduplication based storage optimization technique for virtual desktop [J]. Journal of Computer Research and Development，2012，49(Z1)：125-130.

[2] LIU Jian，CHAI Yunpeng，QIN Xiao，et al. PLC-cache：endurable SSD cache for deduplication-based primary storage [C]// Proceedings of the 2014 30th Symposium on Mass Storage Systems and Technologies (MSST). Piscataway，NJ，USA：IEEE，2014：1-12.

[3] DEBNATH B，SENGUPTA S，LI J. ChunkStash：speeding up inline storage deduplication using flash

memory [C]// Proceedings of the 2010 USENIX Annual Technical Conference. Berkeley, CA, USA: USENIX Association, 2010: 215-229.

[4] MAO Bo, JIANG Hong, WU Suzhen, et al. SAR: SSD assisted restore optimization for deduplication-based storage systems in the cloud [C]// Proceedings of the 2012 IEEE 7th International Conference on Networking, Architecture, and Storage. Piscataway, NJ, USA: IEEE, 2012: 328-337.

[5] MEISTER D, BRINKMANN A. Dedupv1: improving deduplication throughput using solid state drives (SSD) [C]// Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). Piscataway, NJ, USA: IEEE, 2010: 1-6.

[6] GUPTA A, PISOLKAR R, URGAONKAR B, et al. Leveraging value locality in optimizing NAND flash-based SSDs [C]// Proceedings of the 9th USENIX Conference on File and Storage Technologies. Berkeley, CA, USA: USENIX Association, 2011: 91-103.

[7] CHEN Feng, LUO Tian, ZHANG Xiaodong. CaFTL: a content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives [C]// Proceedings of the 9th USENIX Conference on File and Storage Technologies. Berkeley, CA, USA: USENIX Association, 2011: 77-90.

[8] WANG Chundong, WEI Qingsong, YANG J, et al. NV-Dedup: high-performance inline deduplication for non-volatile memory [J]. IEEE Transactions on Computers, 2018, 67(5): 658-671.

[9] CHEN Zhengguo, CHEN Zhiguang, XIAO Nong, et al. NF-Dedupe: a novel no-fingerprint deduplication scheme for flash-based SSDs [C]// Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC). Piscataway, NJ, USA: IEEE, 2015: 588-594.

[10] 王龙翔，董小社，张兴军，等. 内容分块算法中预期分块长度对重复数据删除率的影响 [J]. 西安交通大学学报，2016，50(12)：73-78.
WANG Longxiang, DONG Xiaoshe, ZHANG Xingjun, et al. Influence of expected chunk size on deduplication ratio in content defined chunking algorithm [J]. Journal of Xi'an Jiaotong University, 2016, 50(12): 73-78.

[11] MUTHITACHAROEN A, CHEN Benjie, MAZIÈRES D. A low-bandwidth network file system [C]// Proceedings of the 18th ACM Symposium on Operating Systems Principles. New York, USA: ACM, 2001: 174-187.

[12] ESHGHI K, TANG H K. A framework for analyzing and improving content-based chunking algorithms [EB/OL]. [2020-05-01]. http://www.hpl.hp.com/techreports/2005/HPL-2005-30R1.pdf.

[13] KRUUS E, UNGUREANU C, DUBNICKI C. Bimodal content defined chunking for backup streams [C]// Proceedings of the 8th USENIX Conference on File and Storage Technologies. Berkeley, CA, USA: USENIX Association, 2010: 239-252.

[14] WEN X, DAN F, YU H, et al. FastCDC: a fast and efficient content-defined chunking approach for data deduplication [C]// Proceedings of the 2016 USENIX Annual Technical Conference. Berkeley, CA, USA: USENIX Association, 2016: 101-114.

[15] RABIN M O. Fingerprint by random polynomials: TR-15-81 [R]. Cambridge, USA: Havard University, 1981: 1-6.

[16] QUINLAN S, DORWARD S. Venti: a new approach to archival data storage [C]// Proceedings of the 1st USENIX Conference on File and Storage Technologies. Berkeley, CA, USA: USENIX Association, 2002: 89-101.

[17] BRODER A Z. On the resemblance and containment of documents [C]// Proceedings of the Compression and Complexity of Sequences. Piscataway, NJ, USA: IEEE, 1997: 21-29.

[18] Linux Kernel Organization. The Linux kernel archives [EB/OL]. [2020-05-20]. http://kernel.org/.

[19] ELESAWY M. Real life violence situations dataset [EB/OL]. [2020-05-01]. https://www.kaggle.com/mohamedmustafa/real-life-violence-situations-dataset.

[20] PLUMMER B A, WANG Liwei, CERVANTES C M, et al. Flickr30k entities: collecting region-to-phrase correspondences for richer image-to-sentence models [C]// Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). Piscataway, NJ, USA: IEEE, 2015: 2641-2649.

[21] ROBICQUET A, ALAHI A, SADEGHIAN A, et al. Forecasting social navigation in crowded complex scenes [EB/OL]. [2020-05-04]. https://arxiv.org/abs/160100998.

[22] DENG J, DONG W, SOCHER R, et al. ImageNet: a large-scale hierarchical image database [C]// Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ, USA: IEEE, 2009: 10836047.

（编辑　陶晴）