doi:10.3969/j.issn.1001-4616.2020.02.020

一种基于源端数据重删的数据备份和恢复系统设计与实现

王兴虎[1],何安元[2]

(1.南京航空航天大学计算机科学与技术学院,江苏 南京 211106)
(2.南京航空航天大学信息化技术中心,江苏 南京 211106)

[**摘要**] 针对当前源端数据重删技术中,数据重删效率低、计算指纹耗时长以及频繁操作数据库耗时的问题,设计并实现了一种采用基于源端数据重删技术的数据备份与恢复系统. 系统通过在客户端预先将数据流进行分段并采用预处理环形队列进行存储,基于可变分块对数据段进行分块,整个处理过程并发执行,因此该预处理并发计算模块有效缩短了计算时间,而服务端通过用容器存放临近的数据块与索引信息,设计以容器为单位的多级缓存,明显提高缓存命中率. 此外,通过使用布隆过滤器与多级缓存减少了数据库的操作频率. 仿真实验表明该系统能有效提高数据备份与恢复效率.

[**关键词**] 源端数据重删,备份与恢复效率,预处理并发计算,多级缓存,数据备份与恢复系统
[**中图分类号**]TP311　[**文献标志码**]A　[**文章编号**]1001-4616(2020)02-0131-09

Source Deduplication-based Data Backup and Recovery System

Wang Xinghu[1],He Anyuan[2]

(1.College of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China)
(2.Informationization Technology Center,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China)

Abstract:In order to solve the existing problems of the current source deduplication technology,which is low efficiency of data deduplication,time-consuming fingerprints calculation and frequent requests to the database operation,we design a source deduplication-based data backup and recovery system in this paper. By pre-segmenting the data stream,applying the pre-processed circular queue for storage and segmenting the data block by content-defined chunking at the client,the entire processing process is executed concurrently. The pre-processing concurrent computing module effectively shortens the calculation time. The server stores the adjacent data block and index information by the container and the multi-level caches. The container and multi-level caches are designed in the unit of container,which obviously improves the cache hit rate. Furthermore,the frequent access to the database is optimized by using Bloom filters and multi-level caches. Experimental results show that the system can effectively improve the efficiency of data backup and recovery.

Key words:source data deduplication,efficiency of backup and recovery,pre-processing concurrent computing,multi-level caches,data backup and recovery system

近年来,信息技术的快速发展和传统行业的信息化转型加快导致了数据量的飞速增长. 数据占用了越来越多的存储空间,企业的数据管理和存储成本逐年上升,大量的存储花费与严峻的数据安全问题给传统企业带来了巨大的经济压力. 文献[1]研究报告表明,47%的企业主管担心企业的预算被用于不必要的存储,42%的主管担心高昂的存储成本. 文献[2]研究表明,日常存储的业务数据中存在 60%的冗余业务数据,冗余数据的存在造成了存储资源的浪费. 为了解决数据冗余问题,重复数据删除技术[3-4]成为了近年来的研究热点. 重复数据删除技术可以有效地删除数据存储中的冗余数据,确保数据存储中同样的数据信息只有一份,大大减少了备份时间和存储空间,在备份、归档和数据容灾恢复方面[5]应用广泛. 当前在国际上比较知名的数据备份与恢复产品有基于源端数据的 EMC Avaram,基于目标端的 Data Domain、IBM 的 Diligent 以及 Commvault 的 Simpana.

收稿日期:2019-11-20.
通讯作者:王兴虎,博士研究生,工程师,研究方向:软件工程、计算机网络. E-mail:tiger@ nuaa.edu.cn

重复删除技术会消耗系统大量的计算资源,为了有效提高重删系统的性能,学者提出了一些有效的解决办法:如避免重删系统中的磁盘瓶颈技术[6]以及基于相似的搜索技术[7]、基于集群的增量压缩技术[8]. Sun 等[9]设计了基于分布式系统的重删方案,即解决 NP-Hard 问题. 设计新的文件分区算法,通过增量来记录有效访问,Chu 等[10]使用分块技术减少了相同块的重复对比,提出了 Dis-Dedup 的分配策略;Xue Yua 等[11]在多域架构的基础上提出了云存储中的重删技术 EPCDD. 在综合参考现有技术的基础上,本文设计和实现了基于源端数据重删技术的数据备份与恢复系统,通过在客户端建立预处理并行计算模块和在服务端建立高效的缓存模型,有效地提高整体备份效率.

1　备份与恢复系统结构

根据文件系统执行源端重复数据删除的存储节点位置的不同,目前主要有源端重复数据删除技术和目标端重复数据删除技术. 源端重复数据删除直接在文件系统内实现,少部分的数据通过网络传输;目标端重复数据删除中的所有数据需要通过网络传输到远端存储节点进行重删操作,它会占用大量网络带宽. 在现有重删技术中,基于源端数据重删技术的数据备份具体过程为:根据分块算法对数据流进行分块,然后对分好的块计算哈希(Hash)指纹[12],即对每个数据块生成检索指纹,用来标识其唯一性;把指纹发送服务端进行比对,在已存在的数据库指纹索引表中查找确认,确定数据块是否已经存在备份设备中,根据比对的结果把新数据发送到服务端保存起来,已有的数据就不再发送,以达到节省带宽并节省存储的目的. 在基于源端数据重删的技术中,由于客户端的分块、计算指纹需要大量时间,服务端存放数据时,指纹离散高,频繁操作数据块耗时大,从而整体流程耗时较高. 为了减少耗时,本文提出一个基于源端数据重删的备份与恢复系统,系统的结构如图 1 所示.

该数据备份与恢复系统包含客户端与服务端两部分,客户端主要包括数据流预处理、数据分块、计算指纹与索引信息建立,其中预处理模块可将输入数据流提前分段并存储,便于之后的并行计算. 服务端包括布隆过滤器(Bloom filter)[13-14]、多级缓存模型、数据库与容器(Container). 布隆过滤器与多级缓存模型负责存储容器,对比客户端传来的指纹,便于减少访问数据库的频率. 容器负责存储数据块与索引信息,提高缓存工作的效率. 服务端负责指纹对比、数据块的存储与检索功能.

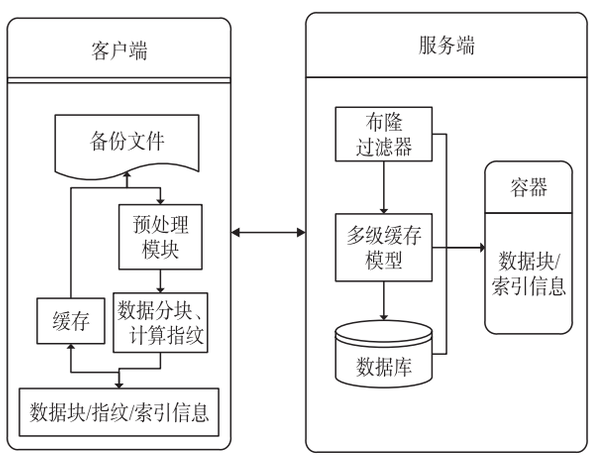

图 1　系统总体结构图

Fig. 1　Structure of backup and recovery system

客户端与服务端之间通过 SAN 网络传输数据块、索引信息与指纹. 在备份过程中客户端将指纹发送至服务端进行对比,依据对比结果将没有指纹的数据块与索引信息送至服务端备份,服务端将接收到的数据块与指纹按位置依次存放在容器中. 恢复数据时,客户端通过读取索引文件把索引信息发至服务端,服务端依据索引信息找到数据块返回到客户端.

该体系结构中,客户端利用数据流分段并对每段数据并发执行分块和指纹计算,服务端用独立线程进行存储数据,并借助布隆过滤器与多级缓存模型,有效缩短时间,提高备份效率.

2　客户端设计

客户端包含预处理模块、数据分块、指纹计算与索引信息建立、缓存. 在备份过程中客户端主要实现重删功能,将新增数据块发送至服务端,并更新数据写入结果. 在恢复过程中,客户端通过读取索引文件中对应数据块的索引信息发送至服务端并将服务端发送来的数据块放在缓存中,重复整个流程直到恢复所有对应的数据块.

2.1　预处理模块

预处理模块主要是先通过对数据流进行分段得到多个数据段,再并行处理多个数据段,对每个数据段

进行分块,并计算每个数据块的指纹.

(1)数据流分段

将待备份的文件以数据流的方式传输到客户端,客户端对数据流可以根据需求设定分段大小,例如以20MB 为标准对数据流分段,即每个数据段为 20MB,尾段可能不满 20MB. 分段的目的是为了方便之后的并行计算.

(2)并行处理

为了实现对多个数据段进行并行处理,在客户端建立一个预处理环形队列,此预处理环形队列用来存储数据段. 将数据段存入预处理环形队列的具体存放过程为:若预处理环形队列中有空间存放传进来的数据段,则将此数据段按顺序存放在预处理环形队列的相应位置;若预处理环形队列中没有足够的空间存放,则等待预处理环形队列中存放的数据段处理完、释放空间后再把传入的数据段存入.

预处理环形队列中每个元素即是一个数据段,每个数据段有各自独立的线程,对队列中所存放的数据段进行并行处理,队列长度可以根据客户端的 CPU 并行计算能力进行配置,并行处理多个数据段可充分利用 CPU 的性能,提高指纹计算的整体性能.

2.2 数据分块

数据分块是数据重删的关键技术,一般来说对数据流的分块技术主要有:定长分块[15]技术(Fix-sized partition,FSP)、变长分块[16]技术(Content-defined chunking,CDC)、滑动块切分[17]技术. 在 FSP 技术中,提供一个预先定义好的块,然后所有文件依据定义好的块的大小进行划分,通过哈希算法给划分后的数据块一个指纹值,再将指纹值与存储中的块指纹值对比,删除相同的值. FSP 技术能够有效减少数据占用的存储空间,降低通过网络传输的数据量,但是处理插入与删除问题效率很低. CDC 技术通过计算窗口 Rabin 指纹[18],将文件划分成长度不同的块,再根据文件内容进行块的划分. CDC 算法流程如图 2 所示.

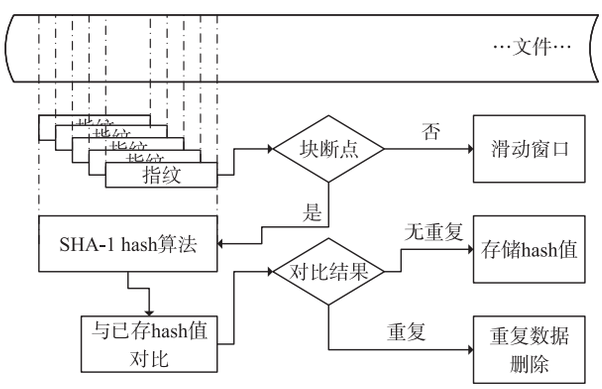

图 2 CDC 算法流程图

Fig. 2 Flow chart of CDC algorithm

从数据流头开始,滑动窗口中的数据流根据 Rabin 算法分成若干数据块,图 2 中垂直虚线是块的边界,当数据块的长度大于预设的最大数据块或者滑动窗口中的 rabin 指纹值满足了预设要求时,把此时窗口位置作为块的边界. 然后划分出的每个块用 hash 函数(MD5[19],SHA-1[20])计算指纹值并与已经存储的数据块对比,如果出现相同的指纹值,就删除代表的数据块,否则存储新的数据块. 滑动窗口大小固定,每次向后滑动一个字节,直到达到预定要求为止. CDC 算法可以有效地提高重删率,重删率越高,节省的磁盘空间越大. 缺点是变长分块计算相对定长分块相对耗时,并且正常变长分块对于数据流都是顺序分块,因为每个数据块的长度不固定,在不破坏每一块的情况下无法从多个位置用不同的线程去分块. 滑块分块技术结合了固定分块与可变分块技术的优点. 首先固定块的大小,用校验函数(Checksum)计算分块后数据块的弱校验值. 弱校验值匹配先前的值,如果匹配不成功则滑动数据块,如果匹配成功则进一步采用 SHA-1 算法对块进行哈希计算得到强校验值,若强校验值匹配成功,则认定该数据块重复;若不匹配则继续滑动数据块.

研究表明[16],当数据量较大时,CDC 算法具有较好的表现,故在本系统中采用 CDC 分块. 由于数据流通过预处理模块已经分段存储在环形队列中,因此本系统在采用 CDC 分块的同时实现了在不破坏每一块的情况下从多个位置用不同线程分块.

2.3 指纹计算与索引信息

数据指纹用来唯一标识数据块的特征,一般选择抗冲突加密 Hash 值作为特征,目前 MD5、SHA1 和 Rabin 是应用最广泛的 HASH 算法,具有计算速度快、节省存储空间的优势,本文采用 Rabin 算法.

(1)指纹计算

Rabin 算法是基于对系数为 Z_2 的不可约多项式进行模运算的算法. 令 $\boldsymbol{S}=[a_1,a_2,\cdots,a_n]$,$\boldsymbol{S}$ 为二进

制字符串,$\boldsymbol{S}(t)$是与字符串 $\boldsymbol{S}$ 关联的 $\boldsymbol{n}-1$ 项多项式,多项式系数为 Z_2,有

$$S(t)=a_1t^{n-1}+a_2t^{n-2}+\cdots+a_{n-1}t+a_n. \tag{1}$$

令 $P(t)=bt^k+b_2t^{k-1}+\cdots+b_kt+b_{k+1}$为 Z_2 的 k 阶不可约多项式,在确定 P 的形式后,可将 Rabin 指纹计算的形式定义为

$$r(t)=S(t)\bmod p(t), \tag{2}$$

定义连续字符串$[X_1\ X_2,\cdots,X_\omega,\ X_{\omega+1},\ X_{\omega+2},\cdots]$,其中每个字符串为 8 位的二进制字符. 使用宽度大小为 ω 的滑动窗口. 假设窗口中字符起点为 X_i,由多项式 $X_i(t)$ 表示. 整个连续字符串$[X_1\ X_2,\cdots,X_\omega,\ X_{\omega+1},\ X_{\omega+2},\cdots]$在窗口中的 Rabin 指纹值为

$$r_i(t)=\left(\sum_{j=1}^{\omega}X_{i+j-1}(t)t^{8\omega-j}\right)\bmod p(t), \tag{3}$$

其中每一 r_i 对应 1 个块. 块由 B_i 表示,$i=1,2,\cdots,n$. 则数据块 B_i 的指纹值$f(B_i)$为

$$f(B_i)=\mathrm{Hash}(B_i). \tag{4}$$

(2)索引信息

索引信息由按顺序记录每个数据块的起始位置、长度和指纹信息组成,供数据恢复时查找.

2.4 缓存

在数据恢复过程中,根据索引信息整理好需要从服务端获取数据块的指纹,在客户端建立一个 16MB 的缓存,用于缓存数据,缓存的大小可以根据具体情况确定,这么做的目的是为了防止每次读取较小的数据时客户端频繁地向服务端要数据,从而在一定程度上减少访问数据库的频率.

3 服务端设计

服务端主要有布隆过滤器、多级缓存模型、数据库与容器,在数据备份过程中服务端负责接收客户端传来的指纹与数据库中存储的指纹进行比对,并存储客户端传来的数据块与索引信息. 在数据恢复过程中,服务端根据客户端发来的索引信息查找相应数据块,并将数据块按照索引顺序拼接起来返回给客户端.

3.1 布隆过滤器

布隆过滤器本质上为概率型数据结构,具有高速查询与插入的特点. 使用布隆过滤器的目的是快速过滤不存在的指纹,数据块的指纹在使用 Hash 算法计算时会在布隆过滤器中留下一个标记,如果一个指纹经过 hash 算法计算后在布隆过滤器中没有找到对应的标记,则说明该指纹是一个新的指纹,对应的数据块也是一个新的数据块,如果指纹经过 hash 算法计算后在布隆过滤器中能找到对应的标记,则说明该指纹可能已经存在,需要经过后面的指纹比对过程继续确认.

在布隆过滤器中,存在假阳性的情况,即待索引的指纹虽然不在集合中,但经过 k 个 Hash 函数映射后布隆过滤器所对应的位置可能为 1,从而误认为该指纹已经在集合里了. 布隆过滤器中把发生假阳性的可能性称为假阳性概率(False positive probability),简称为误判率(Error rate). 误判率由以下公式计算得出,即

$$f_{\mathrm{BF}}=(1-(1-1/m)^{kn})^k\approx(1-\mathrm{e}^{-kn/m})^k. \tag{5}$$

集合的指纹个数为 n、位向量长度为 m 和 Hash 函数的个数为 k.

假设 m 和 n 已知,此时令 $q=k\cdot\ln(1-(1-1/m)^{kn})$,则 $f_{\mathrm{BF}}=\mathrm{e}^q$,而且 f_{BF}和 q 极值点相同,再令 $p=(1-1/m)^{kn}$,则可推导出

$$q=\frac{1}{n\cdot\ln(1-1/m)}\ln p\cdot\ln(1-p). \tag{6}$$

由 q 与 p 的对称性可得,$p=1/2$ 是 q 和 f_{BF}取最小值,即布隆过滤器的位数组中 0 和 1 各占一半,原式换算得 $k\approx\frac{m}{n}\ln 2$,此时的最小误判率称为理想误判率.

在本系统中,采用分段型布隆过滤器(Segemented Bloom filter),分段型布隆过滤器的 k 个哈希函数分别映射到 k 个长度为 m/k 的不相交的位段,将布隆过滤器的查询复杂度达到 $O(1)$,假阳性概率为

$$f'_{\mathrm{BF}}=(1-(1-k/m)^{n})^{k}\approx(1-\mathrm{e}^{-kn/m})^{k}. \tag{7}$$

尽管f'_{BF}稍大于f_{BF},由于m与n较大,误差可忽略不计.

3.2 多级缓存模型

多级缓存模型由一级缓存与二级缓存组成. 一级缓存用于同步的更新命中的容器,布隆过滤器中能找到的指纹,需要在指纹比对的后续流程继续确认是否真的存在,如果在一级缓存和二级缓存中都没有找到对应的指纹记录,并且在数据库的指纹表中找到该指纹,那么通过数据库中的记录找到该指纹存放的容器,把容器中的所有指纹更新到一级缓存中. 二级缓存的作用是在一级缓存更新找到容器的同时,找到该容器 id 临近的下一个容器,找到把容器 id 对应的容器,并把该容器内指纹更新到二级缓存中. 布隆过滤器与多级缓存模型一同组成高效缓存模型,如图 3 所示.

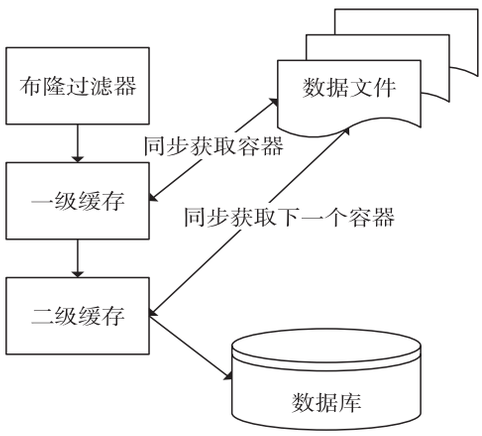

图 3 高效缓存模型

Fig. 3 Model for efficient cache

在高效缓存模型中,每个指纹对比流程为:约定指纹存在则标记为 1(对于标记为 1 的说明数据块已经在服务端有了,客户端不需要再发送),不存在标记为 0. 首先去布隆过滤器里查找,若没有此指纹则标记为 0,流程结束,若有此指纹(根据布隆过滤器特性上面已经说明,布隆过滤器中能查到的指纹不一定存在,需要走后面的流程继续确认)则去一级缓存中查找,一级缓存中若有则标记为 1,流程结束,若没有则进一步去二级缓存中查找,二级缓存中若有则标记为 1,流程结束,二级缓存中若还没有则去数据库中查找,数据库中若还没有,则标记为 0,流程结束,若有则标记为 1,并把该指纹对应的容器同步更新到一级缓存中,下一个容器异步的更新到二级缓存中.

3.3 容器

采用基于流的块排列技术(Stream-Informed Segment Layout,SISL),SISL 技术主要特点是面对同一个数据流,将新数据块存放在一起,块描述符也存放在一起,即数据块在磁盘的物理空间上尽可能地连续存储,并且其对应的指纹也连续存储. 本系统服务端通过设立容器实现 SISL 技术. 服务端先把客户端传来的数据块放到缓存队列,然后顺序放到容器中,这样就能保证临近数据存放的位置也是临近的,以容器为单位的缓存能大大提高缓存的命中率,降低访问数据库的次数,这样在指纹比对和恢复数据时都能有比较高的效率. 容器是一段数据组合的概念,其固定大小为 4MB 的一段数据. 它的数据组织结构是前 24KB 存放指纹及数据块的起始位置和长度信息,从 4MB-24KB 的位置开始存放数据块. 一个容器一般可以放 800 个左右的数据块,由于数据块的长度不固定,因此这个数量也不固定.

当服务端接收到数据块与索引信息时,将每块数据存储至容器的步骤如下:

(1)服务端将传过来的新数据块放到容器中,数据块按照容器中存放数据块的位置依次存放,数据块的指纹按照容器中存放指纹的位置依次存放,并在数据库中记录该指纹对应的容器 id;

(2)容器写满后把容器放到文件中,并在数据库中记录该容器对应的文件 id. 然后创建新的容器,过程为:清空当前容器中的数据(当前容器中的数据已经保存在文件中了),容器 id 加 1,并将容器的信息记录到数据库中.

(3)文件放在磁盘上并在数据库中记录文件对应的磁盘位置. 这样就可以根据数据库中的指纹记录一层层地找到对应的数据块.

系统中每个文件最大为 1 GB,一个数据文件放满了,才会生成一个新的文件存放容器,一个文件可以放 256 个容器. 文件放满容器后会创建新的文件,并把文件的信息记录到数据库中.

4 数据备份与恢复流程

本数据备份与恢复系统克服现有技术中的不足,提供一种基于源端数据重删的数据备份与恢复方法,解决了现有备份与恢复技术中数据的重删效率低、计算指纹耗时长、频繁操作数据库耗时的问题. 下面分

别介绍数据的备份流程与数据的恢复流程.

4.1　数据备份流程

数据备份流程如图 4 所示.

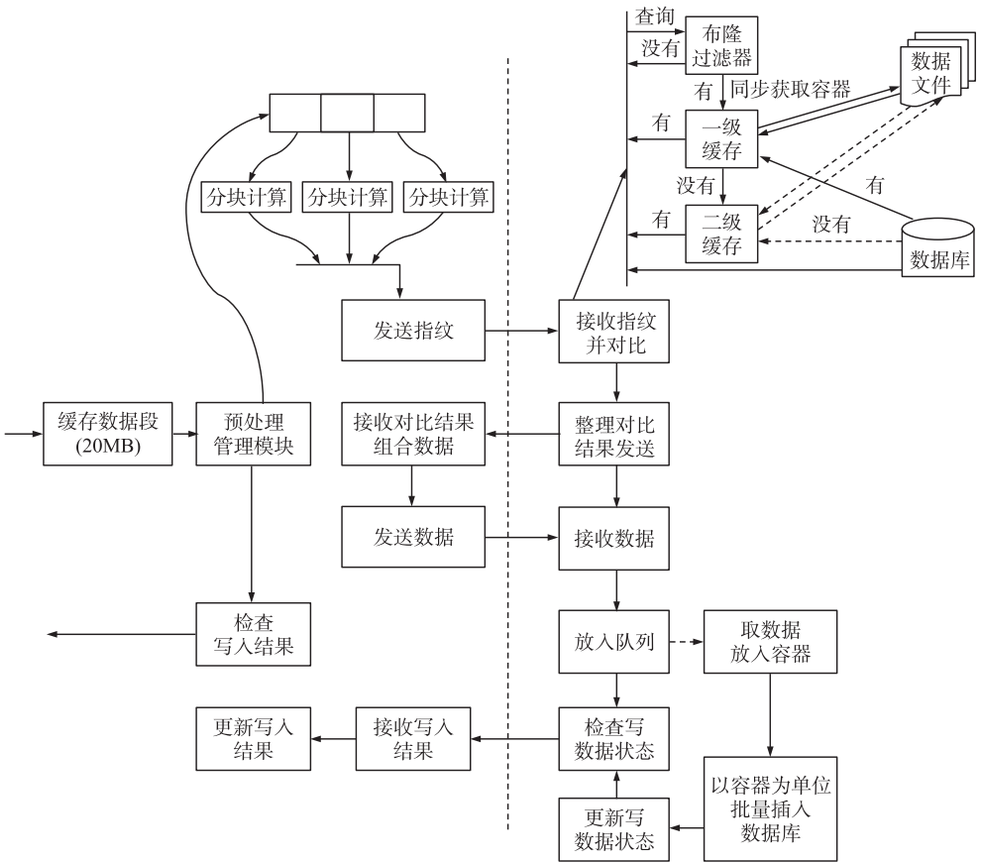

图 4　数据备份流程

Fig. 4　Data-backup flow chart

(1)在客户端,对数据流进行分段得到多个数据段;

(2)并行处理多个数据段,对每个数据段进行分块,并计算每个数据块的指纹;

(3)顺序将指纹发送服务端进行对比,并将对比结果返回至客户端;

(4)客户端根据对比结果,将没有的数据块发送至服务端进行保存备份,服务端将数据块存放状态返回给客户端.

4.2　数据恢复流程

数据恢复流程如图 5 所示.

(1)客户端从索引文件中读取一段待恢复文件的索引,把索引信息发送到服务端.

(2)服务端根据索引信息找到数据块返回客户端,其具体步骤为:解析每个数据块的索引信息,根据索引信息中的指纹先到一级读缓存中查找,若找到则读取数据块,继续找下一数据块;若找不到则去二级读缓存中查找,若二级读缓存中能找到,读取数据块,继续找下一数据块,若找不到则去数据库中找,在数据库中根据指纹找到对应的容器,根据容器 id 找到对应的文件,从文件中读出对应的容器更新到一级读缓存中,并把对应容器的下个容器异步更新到二级读缓存中;将读到的每个数据块按照索引顺序拼接起来返回给客户端.

(3)循环执行以上两步直到获取文件所对应的所有数据块,恢复出完整的文件.

恢复过程中用到一级读缓存和二级读缓存和备份过程中用到一级缓存和二级缓存逻辑上类似,不同的地方在于在备份过程中用到的一级缓存和二级缓存只需要缓存指纹即可,而恢复过程中用到一级缓存和二级缓存除了缓存指纹还要缓存指纹对应的数据块. 恢复中用的缓存和备份中用的缓存是各自独立的.

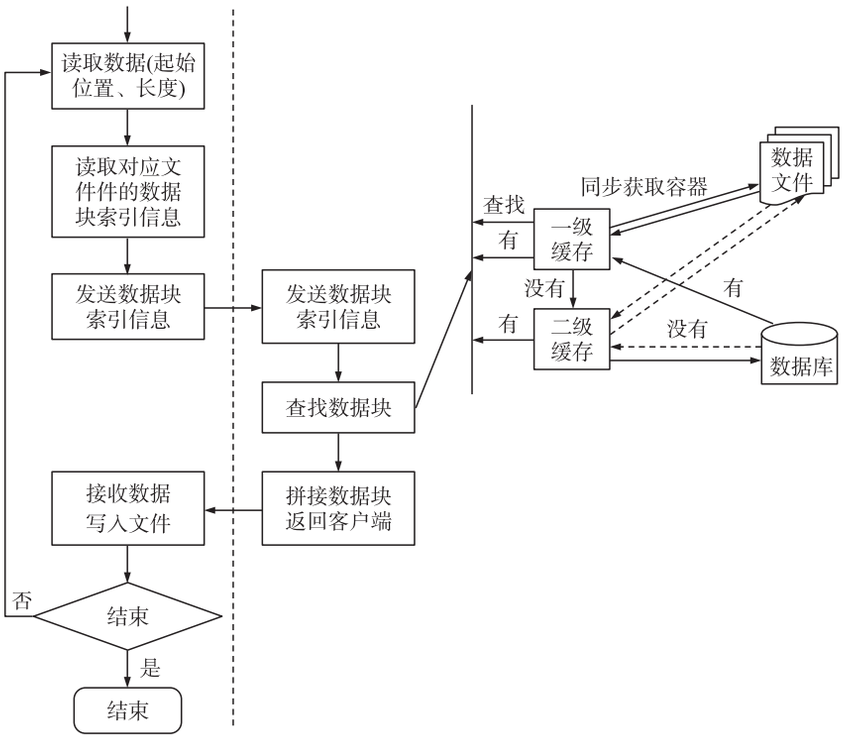

图 5　数据恢复流程图

Fig. 5　Data recovery flow chart

5　系统仿真实验

本系统为基于源端的数据重删与备份系统,系统采用 C/S 架构,基于 Java 语言. 在系统仿真实验中,建立测试服务器与测试机进行多线程的备份与恢复测试,测试服务器配置如表 1 所示,测试机配置如表 2 所示.

表 1　测试服务器的配置

Table 1　Configuration of test server

名称	规格型号
主板	超威 X10DRL-i
CPU	英特尔 E5-2603V4
内存	16 GB
系统盘	英特尔 240 GB SSD
数据盘	4 TB

表 2　测试机的配置

Table 2　Configuration of test machine

名称	规格型号
操作系统	Windows 2012 R2 Server Standard 64 位
主板	技嘉 Z270-Gaming 3
CPU	Intel Core i7-7700K @4.20 GHz 四核
内存	16G * 3 = 32 GB 金士顿
硬盘	三星 SSD 850 EVO 250 GB(250 GB/固态硬盘)

在 Windows 环境下,对同一个数据样本进行两次文件备份测试,数据样本由 ghost 的镜像文件与 windows 的系统目录文件组成,共 13.2GB. 文件样本分别基于本系统的数据备份恢复产品与传统的数据备份恢复系统中测试,测试备份系统的数据备份速度与重删率,其中重删率为备份集实际大小与存储大小之间差值占实际大小的百分比,即系统在进行重删时删掉的重复数据所占百分比. 备份的实验结果如表 3,4 所示.

表 3　本文数据备份系统实验结果

Table 3　Experimental results for the proposed data backup system

改进数据重删备份恢复系统	第 1 次	第 2 次
备份总时间	4 min 22 s	2 min 38 s
备份集实际大小/GB	13.2	13.2
备份集存储大小/GB	10.88	4.79
平均备份速度/(Mb·s^{-1})	51.59	85.55
重删率	17%	99%

表 4　现有的数据备份系统实验结果

Table 4　Experimental result for the present date backup system

现有数据备份系统	第 1 次	第 2 次
备份总时间	9 min 26 s	2 min 48 s
备份集实际大小/GB	13.2	13.2
备份集存储大小/GB	11.12	13.79
平均备份速度/(Mb·s^{-1})	35	80
重删率	16%	99%

备份速度为源数据除以时间. 改进后的备份系统与现有备份系统的备份速度对比如图 6 所示.

由表中数据可以明显发现,与采用传统源端数据重删技术的现有产品相比,尽管双方在重删率上十分接近(这是由于在重删中当前主要采用 CDC 分块技术,故大家的重删率才会十分接近),但在备份速度方面无论是初次备份还是第 2 次备份,基于本系统的数据备份恢复产品对比现有的数据备份恢复产品在速度上有明显优势. 在数据恢复方面,实验结果如表 5 所示.由表 5 可见,该系统在进行数据恢复时也能保持较高的恢复速度.

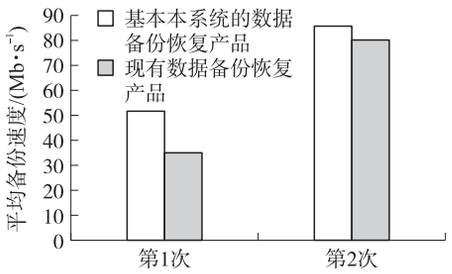

图 6　数据备份速度

Fig. 6　Speed of different data backup system

表 5　数据恢复结果

Table 5　Experimental results for data recovery

	本文的数据备份恢复系统	现有数据备份恢复系统
恢复总时间	3 min 45 s	4 min 2 s
备份集实际大小/GB	13.2	13.2
平均恢复速度/(Mb·s^{-1})	60.07	55.73

6　结论

针对传统基于源端数据重删的数据备份与恢复技术中,因为重删导致耗时过长、服务端存放数据频繁操作数据库耗时所产生的问题,本文设计了一种改进后基于数据重删的数据备份与恢复的系统,对于重删耗时过长的问题,本文通过在客户端对数据流分段并存放至预处理环形队列,实现将重删功能块分布执行、减少重删的耗时,在服务端针对存储数据的耗时问题,设立布隆过滤与多级缓存模型进行指纹对比,减少访问数据库频率,并基于块排列技术,将数据与指纹存放在容器中,提高在对比指纹时指纹的命中率. 结果表明,相比于现有的基于源端数据重删的数据备份与恢复系统,本系统在数据备份与恢复方面具有良好的表现.

[参考文献]

[1] 郭平. 消除冗余解放容量[EB/OL]. http://www.ccw.com.cn/07/0710/c/0710c24_4.html,2007-03-19/2012-06-07.

[2] 云端数据碎片可能增加企业存储成本[EB/OL]. http://news.west.cn/58481.html,2019-07-10/2019-12-01.

[3] 彭刚. 一种面向容灾存储系统的重复数据删除方法[J]. 网络安全技术与应用,2018(4):43-44.

[4] 朱江,冀鸣,杨志成,等. 基于重复数据删除技术的存储系统分析[J]. 信息系统工程,2017(4):70-72.

[5] YANG T M,JIANG H,FENG D,et al. DEBAR:A scalable high-performance de-duplication storage system for backup and archiving[C]//Proceeding of the IEEE IPDPS'10. Piscataway,NJ:IEEE,2010:1-12.

[6] ZHU B,LI K. Avoiding the disk bottleneck in the data domain deduplication file system[C]//Proceeding of the 6th Usenix Conference on File and Storage Technologies. Berkeley:USENIX Association,2008:269-282.

[7] BHAGWAT D,ESHGHI K,MEHRA P. Content-based document routing and index partitioning for scalable similarity-based searches in large corpus[C]//Proceeding of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York:ACM Press,2007:105-112.

[8] OUYANG Z,MEMON N,SUEL T,et al. Cluster-Based delta compression of a collection of files[C]//Proceeding of the 3rd International Conference on Web Information Systems Engineering. Washington:IEEE Computer Society Press,2006:257-266.

[9] SUN Y S. Online data deduplication for in-memory big-data analytic systems[C]//2017 IEEE International Conference on Communications(ICC). [S.l.]:IEEE,2017.

[10] CHU X,ILYAS I F,KOUTRIS P. Distributed data deduplication[J]. Proceedings of the VLDB endowment,2016,9(11):864-875.

[11] YANG X,LU R,CHOO K K R,et al. Achieving efficient and privacy-preserving cross-domain big data deduplication in cloud[J]. IEEE transactions on big data,2017(1):1-3.

[12] BOLOSKY W J,CORBIN S,GOEBEL D,et al. Single instance storage in Windows 2000[C]//Proceeding of the 4th Usenix Windows System Symp. Berkeley:USENIX Association,2000:13-24.

[13] BORDER A Z,MITZENMACHER M. Network applications of bloom filters:A survey[J]. Internet Mathematics,2003,1(4):

485-509.
[14] MITZENMACHER M. Compressed bloom filters[J]. IEEE ACM transaction on networking,2002,10(5):604-612.
[15] BOBBARJUNG D R,JAGANNATHAN S,DUBNICKI C. Improving duplicate elimination in storage systems[J]. ACM Transaction on storage,2006,2(4):424-448.
[16] JAIN N,DAHLIN M,TEWARI R. Taper:Tiered approach for eliminating redundancy in replica synchronization[C]// Proceeding of the 4th Usenix Conference on File and Technologies. Berkeley:USENIX Association,2005:281-293.
[17] BORDER A Z. Identifying and filtering near-duplicate documents[C]//Proceeding of the 11th Annual Symp on Combinatorial Pattern Matching. London:Springer-Verlag,2000:1-10.
[18] 刘文龙,李晖,金东勋. 数字指纹生成方案及关键算法研究[J]. 信息网络安全,2015(2):66-70.
[19] DOUGLIS F,IYENGAR A. Application-specific delt encoding via resemblance detection[C]//Proceeding of the 2003 USENIX Anuual Technical Conference. Berkeley:USENIX Association,2003:113-126.
[20] KULKARNI P,DOUGLIS F,LAVOIE J D,et al. Redundancy elimination within large collection within large collections of files[C]//Proceeding of the 2004 Usenix Anuual Technical Conference. Berkeley:USENIX Association,2004:59-72.

[责任编辑:陆炳新]

(上接第 113 页)

[10] DUMONT M,SCHLICHTER J,CARDILLO T,et al. CYC2 encodes a factor involved in mitochondrial import of yeast cytochrome c[J]. Molecular and cellular biology,1993,13(10):6442-6451.
[11] MCALLISTER S,POLSON S,BUTTERFIELD D,et al. Validating the Cyc2 neutrophilic iron oxidation pathway using meta-omics of zetaproteobacteria iron mats at marine hydrothermal vents[J]. mSystems,2020,5(1):553-559.
[12] SANCHEZ N S,PEARCE D A,CARDILLO T S,et al. Requirements of Cyc2p and the porin,Por1p,for ionic stability and mitochondrial integrity in Saccharomyces cerevisiae[J]. Archives of biochemistry and biophysics,2001,392(2):326-332.
[13] MAYR J A,HAACK T B,FREISINGER P,et al. Spectrum of combined respiratory chain defects[J]. Journal of inherited metabolic disease,2015,38(4):629-640.
[14] SOUBANNIER V,VAILLIER J,PAUMARD P,et al. In the absence of the first membrane-spanning segment of subunit 4(b),the yeast ATP synthase is functional but does not dimerize or oligomerize[J]. The journal of biological chemistry,2002,277(12):10739-10745.
[15] SOUBANNIER V,RUSCONI F,VAILLIER J. The second stalk of the yeast atp synthase complex:identification of subunits showing cross-links with known positions of subunit 4(Subunit b)[J]. Biochemistry,1991,38(45):15017-15024.
[16] ZHANG C,WANG R,LIU Z,et al. The plant triterpenoid celastrol blocks PINK1-dependent mitophagy by disrupting PINK1's association with the mitochondrial protein TOM20[J]. The journal of biological chemistry,2019,294(18):7472-7487.

[责任编辑:黄　敏]