ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.5, May 2010, pp.916−929
doi: 10.3724/SP.J.1001.2010.03761

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

重复数据删除技术*

敖 莉[1], 舒继武[1,2+], 李明强[1]

[1](清华大学 计算机科学与技术系,北京 100084)

[2](清华大学 信息科学与技术国家实验室(筹),北京 100084)

Data Deduplication Techniques

AO Li[1], SHU Ji-Wu[1,2+], LI Ming-Qiang[1]

[1](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China)

+ Corresponding author: E-mail: shujw@tsinghua.edu.cn

Ao L, Shu JW, Li MQ. Data deduplication techniques. *Journal of Software*, 2010,21(5):916−929. http://www.jos.org.cn/1000-9825/3761.htm

Abstract: Data deduplication technologies can be divided into two categories: a) identical data detection techniques, and b) similar data detection and encoding techniques. This paper presents a systematic survey on these two categories of data deduplication technologies and analyzes their advantages and disadvantages. Besides, since data deduplication technologies can affect the reliability and performance of storage systems, this paper also surveys various kinds of technologies proposed to cope with these two aspects of problems. Based on the analysis of the current state of research on data deduplication technologies, this paper makes several conclusions as follows: a) How to mine data characteristic information in data deduplication has not been completely solved, and how to use data characteristic information to effectively eliminate duplicate data also needs further study; b) From the perspective of storage system design, it still needs further study how to introduce proper mechanisms to overcome the reliability limitations of data deduplication techniques and reduce the additional system overheads caused by data deduplication techniques.

Key words: network storage system; duplicate data; data elimination; reliability; performance

摘 要: 重复数据删除技术主要分为两类:相同数据的检测技术和相似数据的检测与编码技术,系统地总结了这两类技术,并分析了其优缺点.此外,由于重复数据删除技术会影响存储系统的可靠性和性能,又总结了针对这两方面的问题提出的各种技术.通过对重复数据删除技术当前研究现状的分析,得出如下结论:a) 重复数据删除中的数据特性挖掘问题还未得到完全解决,如何利用数据特征信息有效地消除重复数据还需要更深入的研究;b) 从存储系统设计的角度,如何引入恰当的机制打破重复数据删除技术的可靠性局限并减少重复数据删除

* Supported by the National Natural Science Foundation of China under Grant No.60873066 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2009AA01A403 (国家高技术研究发展计划(863)); the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No.200800030027 (高等学校博士学科点专项科研基金)

Received 2008-06-04; Revised 2008-12-29; Accepted 2009-10-09

技术带来的额外系统开销也是一个需要深入研究的方面.

关键词: 网络存储系统;重复数据;数据删除;可靠性;性能

中图法分类号: TP311 **文献标识码**: A

随着数字信息量的爆炸式增长,数据占用空间越来越大;在过去的 10 年里,很多行业提供的存储系统容量从数十 GB 发展到数百 TB,甚至数 PB[1].随着数据的指数级增长,企业面临的快速备份和恢复的时间点越来越多,管理保存数据的成本及数据中心空间和能耗也变得越来越严重.研究发现,应用系统所保存的数据中高达60%是冗余的,而且随着时间的推移越来越多[1].为了缓解存储系统的空间增长问题,缩减数据占用空间,降低成本,最大程度地利用已有资源,重复数据删除技术已成为一个热门的研究课题.一方面,利用重复数据删除技术可以对存储空间的利用率进行优化,以消除分布在存储系统中的相同文件或者数据块.另一方面,利用重复数据删除技术可以减少在网络中传输的数据量,进而降低能量消耗和网络成本[2],并为数据复制大量节省网络带宽.

随着重复数据删除技术的发展,该技术大量应用于存储备份和归档系统中,该系统中的重复数据删除模块负责对数据内容进行比对分析,查找出冗余数据,并将其元数据反馈给存储服务接口,最后将不重复的数据存入到存储介质中.重复数据删除技术主要分为以下两大类:

(1) 相同数据检测技术.相同数据主要包括相同文件及相同数据块两个层次.完全文件检测(whole file detection,简称 WFD)技术主要通过 hash 技术[3]进行数据挖掘;细粒度的相同数据块主要通过固定分块(fixed-sized partition,简称 FSP)检测技术[4]、可变分块(content-defined chunking,简称 CDC)检测技术[4−7]、滑动块(sliding block)技术[8]进行重复数据的查找与删除.

(2) 相似数据检测和编码技术.利用数据自身的相似性特点,通过 shingle 技术[8−12]、bloom filter 技术[6,13−15]和模式匹配技术[16,17]挖掘出相同数据检测技术不能识别的重复数据;对相似数据采用 delta 技术[10,11,18,19]进行编码并最小化压缩相似数据,以进一步缩减存储空间和网络带宽的占用.

上述这些技术使得共享数据块的文件之间产生了依赖性,几个关键数据块的丢失或错误可能导致多个文件的丢失和错误发生,因此它同时又会降低存储系统的可靠性,为此,一些研究者又引入了冗余复制技术[7,20]和纠删码技术[12]等来提高重复数据删除系统的可靠性.另外,因数据的检测对比等过程导致大量的计算开销,重复数据删除技术对存储系统的性能影响也很大,为此,一些研究者提出了一些关键技术,如减轻磁盘瓶颈技术[21]、提高数据搜索速度的技术[22]和提高相似数据编码速度的技术[19].

基于目前的研究现状,还有一些需要深入研究的方面,比如,如何充分挖掘数据的内在特性,提出更为高效的重复数据删除技术,如何提高重复数据删除系统的可靠性,如何提高重复数据删除技术的性能,如何融合各种现有技术,提高重复数据删除技术的通用性、可扩展性和自适应性,等等.

本文第 1 节讨论相同数据检测的各项技术,并分析各技术的优缺点和适用性.第 2 节首先讨论用于相似数据检测的各项技术及其特点,然后介绍对相似数据进行编码的 delta 技术.第 3 节介绍目前重复数据删除技术的可靠性研究,并分析每种技术的特点.第 4 节讨论重复数据删除技术的性能研究,分析为平衡系统空间和时间开销所采用的一些技术方案及特点.第 5 节对本文进行总结.

1 相同数据检测技术

相同数据检测技术是将数据进行划分,找出相同的部分,并且以指针取代相同数据的存储.

1.1 完全文件检测技术

WFD 技术是以文件为粒度查找重复数据的方法.如图 1 所示,首先对整个文件进行 hash 计算,然后将该值与已存储的 hash 值进行比较,如果检测到相同的值,则仅将文件用指针替换,不进行实际存储,否则存储新文件.

研究者将 hash 算法引入到重复数据删除技术中,是利用了 hash 值可以唯一地表征特定的数据实体,通过 hash 技术,数据被标志为一个固定大小的值,比较该值,就可以判别数据的重复性.目前 MD5 和 SHA1 是应用最

广泛的 Hash 算法[10,11],两者的抗冲突性都比较好.

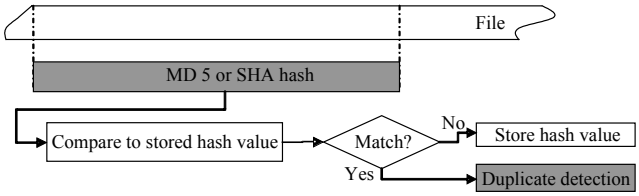


Fig.1 WFD technique

图 1 WFD 技术

Windows 2000 的 SIS(single instance storage)[3]应用该技术对具有 20 个不同 Windows NT 映像的服务器进行测试,结果表明总共节省了 58%的存储空间.

基于 hash 算法的完全文件检测技术具有两个优势:(1) 在普通硬件下计算速度很快,加州大学研究表明[23], SHA-1 是 83MB/S,而 MD5 是 227MB/S;(2) 可以检测到所有完全相同的文件,节省存储空间较大.但是,该方法也有两个主要缺点:(1) 对于较大的数据集,需要比较的范围大,耗时多;(2) 不能检测不同文件内部的相同数据.

1.2 基于FSP算法的块检测技术

完全文件检测技术不能用于文件内部的重复数据查找,因此有研究者提出了更细粒度——块级别的重复数据检测.基于固定尺寸划分算法(FSP)的相同数据块检测技术是使用固定大小的分块策略在存储系统中识别相同数据的方法,如图 2 所示.该方法分 3 个步骤:(1) 提供一个已经预先定义好的块的大小(该值独立于所存取的数据内容),所有文件均按照这个固定的块大小进行划分[4];(2) 每个划分好的数据块均通过哈希算法(MD5 或 SHA1)得到一个指纹值;(3) 将该值与已存储的块指纹值进行比对,如果检测到相同的值,则删除其代表的数据块,否则存储新的数据块.

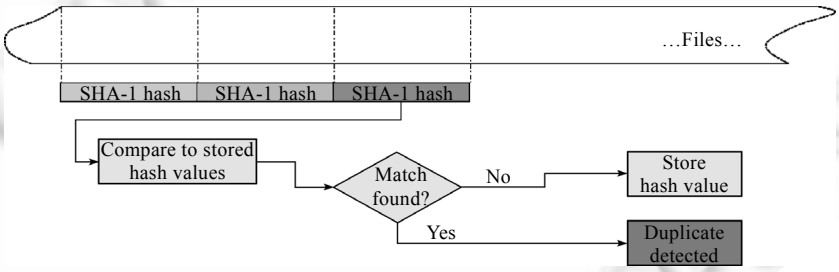


Fig.2 Identical data chunk detection technique based on FSP algorithm[7]

图 2 基于 FSP 算法的相同数据块检测技术[7]

基于 FSP 算法的相同数据块检测技术已应用在很多领域,并具有如下两个特征:(1) 缩减存储空间,如针对数据归档的网络存储系统 Venti[24]应用该技术减少了大约 30%的存储空间;(2) 减少网络传输的数据量,如虚拟机优化系统[25]通过该技术加速了在低带宽网络上的数据传输并改进了内存的性能.

此外,基于 FSP 算法的相同数据检测技术还可以提供很高的处理速度,适合于在交互性的环境中应用[5].但是它也具有一定的局限性:对编辑和修改的序列很敏感,对于插入问题(在原来的数据流中某处插入少量新字节,其他部分不变)和删除问题(在原来的数据流中某处删除少量新字节)处理十分低效,不能智能地根据文件自身内容的变化和文件之间的关联关系进行调整与优化,基于此,研究者们提出了可变块大小划分的检测技术.

1.3 可变分块检测技术

1.3.1 基于 CDC 算法的检测技术

CDC 算法是应用 Rabin 指纹将文件分割成长度大小不一的分块策略[4,20].与固定分块策略不同的是,它对文

件进行块划分的方法是基于文件内容的,因此数据块大小是可变的,如图 3 所示,其过程有两步:(1) 一个文件按照 CDC 算法分为若干数据块.图 3 显示了一个数据流或一个文件以及由虚垂直线表示的块边界.CDC 算法首先从文件头开始,将固定大小(互相重叠)的滑动窗口中的数据看成组成文件的各个部分.在窗口的每个位置,该窗口中数据的一个指纹(Fingerprint)被计算出来.在实际中,鉴于 Rabin 指纹[26]计算的高效性及 Rabin 指纹函数的随机性(对任意数据呈现出均匀分布),研究者们多使用其计算滑动窗口内容的指纹值.当指纹满足某个条件时,如当它的值模某个指定的整除数为 0 时,则把此时窗口的位置作为块的边界.重复这个过程,直到整个文件数据都被分为块.(2) 划分出的每个块用 hash 函数(MD5[27],SHA-1[28]或更高的 SHA 标准[29]函数)计算出它的指纹值并与已存储的数据块进行对比,如果检测到相同的指纹值,则删除其代表的数据块,否则存储新的数据块.

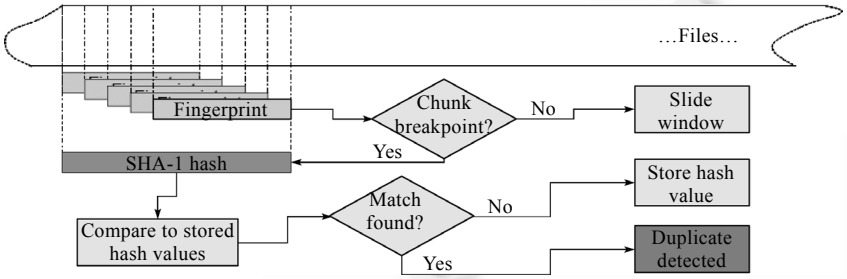


Fig.3 Identical data chunk detection technique based on CDC algorithm[7]

图 3 基于 CDC 算法的相同数据块检测技术[7]

当前可变分块检测技术已应用在 P2P 文件系统 Pasta[30]、Pastiche 备份系统[31]、基于值的网络缓存系统[32]、Deep Store 归档存储系统[23]和低带宽网络系统 LBFS[2]中.在 LBFS 中,系统针对数据分块可能出现的病态现象(滑动窗口数据的指纹值不能符合条件,因此块的边界不能确定,导致块过大)进行了改进,将块大小进行了限定,给出数据大小分块的上、下限.

综上,根据 CDC 算法的特性,无论是插入还是删除一小部分字节,都只会影响一到两个块,其余的块保持不变,即 CDC 方法在两个相似对象(只相差几个字节)之间可以检测出更多的冗余.

但是,CDC 算法也存在一定的局限性,它划分的粒度绝大部分取决于期望块的设定.如果该值设置得较小,那么,虽然粒度较细,重复数据查找较为精确,但是额外存储每块信息的开销很大;反之,如果该值设置得较大,则粒度过粗,重复数据删除的效果不好.所以,如何权衡精确查找和额外开销是一个难点.

1.3.2 基于 fingerdiff 算法的检测技术

为了弥补 CDC 算法额外存储空间开销大的缺陷,研究者提出了 fingerdiff 算法[4],其核心思想是将没有变化的块尽可能地合并,以减少各个数据块的元数据占用的存储空间.应用 fingerdiff 算法进行相同数据块检测过程包含 3 个步骤:(1) 一个文件按照 CDC 算法进行子块**的划分.(2) 每个子块按照 fingerdiff 设置的最大子块数进行合并.(3) 每个块用 hash 函数计算出它的指纹值,然后对比已存储的数据块指纹值,如果检测到相同的指纹值,则删除其对应的数据块;否则将大块进行拆分,找到最小的不同数据块进行存储,其余块仍然保持合并状态.

以上 CDC 算法和 fingerdiff 算法的检测技术存在两个不足之处:(1) 在可变分块检测过程中,对于一个文件对间较小的随机改变,效果不好,两种算法的适应性很差;(2) 两种技术中数据块的划分都是根据内容可变的,但该值在很大程度上取决于算法中期望块大小的设定,而期望块大小的选择依赖于文件相似性程度和文件修改的位置,这对相同数据检测的性能影响很大.因此,可以通过深入挖掘数据特征规律,自适应地调整数据块的大小,选择符合数据特征和性能指标的最佳块大小.

** 子块是组成块的单位.在 fingerdiff 算法中用 CDC 算法划分出的块称为子块,之后子块合并生成的块称为块.

1.4 滑动块检测技术

滑动块检测技术结合了固定块大小检测技术和可变块大小检测技术的优点,块大小固定,管理简单[4].文献[6]通过测试发现,对大的簇,CDC 的重复数据检测性能较好,而滑动块技术对细粒度匹配更适用.如图 4 所示,基于滑动块技术的相同块检测过程有 4 步:(1) 一个文件用 rsync 求和校验(checksum)函数[33]和固定块大小的滑动窗口来计算文件对象的每个重叠块的求和校验值.(2) 对于每个块,比较求和校验值与先前存储的值.(3) 若匹配,则利用更严格的 SHA-1 算法对块进行 hash 计算,并将 SHA-1 hash 值与先前存储的值进行比较,从而进行冗余检测.如果检测到数据冗余,将其记录后,滑动窗口越过这个冗余块继续前移.而且先前已经被划分的块和最近被检测到冗余之前的这个碎片需要被记录并且存储.(4) 如果求和校验值或 hash 值不能匹配,则滑动窗口继续前移.如果滑动窗口已经移动了一个块大小的距离,但是仍然无法匹配到任何已经被存储的块,则需要对这个块进行求和校验和 SHA-1 hash 计算,并存储在各自的表中,用作以后块的比较对象.

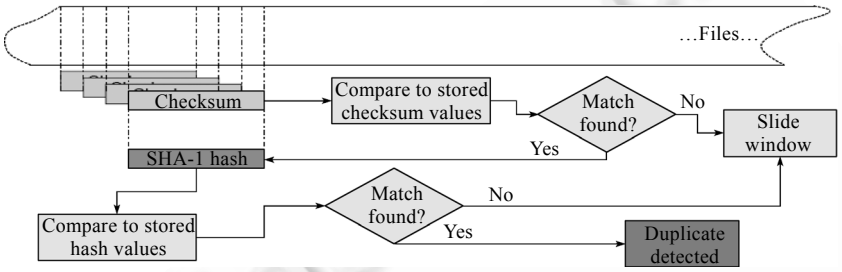


Fig.4 Identical data chunk detection technique based on sliding blocking method[7]

图 4 基于滑动分块方法的相同数据块检测技术[7]

滑动块检测技术的特点是对插入问题和删除问题处理高效,并能检测到更多的冗余数据.(1) 插入问题:如果一小部分字节被插入到一个对象中,则只有周围的块会改变,其他的块仍将被识别出来并被匹配,并且一个长度等于插入的字节数的碎片会产生出来.(2) 删除问题:删除一小部分字节也会产生一个长度等于块大小减去被删除部分字节长度的碎片,其他块不受影响.但滑动块检测技术也存在一个不足:在插入和删除问题中都会引入碎片,如何能够准确识别改变的数据,不影响匹配数据块,从而少产生额外的碎片将是一个研究的难点.

1.5 小 结

本节介绍了相同数据检测的 5 种技术,重复数据的存在形式中有很大一部分是完全相同的数据.文献[5]采用具有不同相关性的数据集评估了完全文件(WFD)、固定分块(基于 FSP 算法)、可变分块检测技术(基于 CDC 算法)检测相同数据的效果,并给出了在变化的数据类型上使用这些方法所获得的效果.如图 5 所示,针对不同的测试集,WFD 技术、基于 FSP 的检测技术、基于 CDC 的检测技术在数据类型比较分散的数据集上检测效果都不佳,仅检测到 17%~30%的相同数据,而在有潜在关联关系的数据集中,基于 CDC 的检测技术可检测到 60%左右的相同数据,但 WFD 技术和基于 FSP 的检测技术仅检测到 20%~30%的相同数据.

同时,文献[7]针对块级别的相同数据检测技术,在 IBM 实验室真实数据集上测试,给出了基于 FSP 的检测技术、基于 CDC 的检测技术和滑动块检测技术(sliding block)的性能比较,如图 6 所示,针对 6 种不同测试集,Sliding block 技术在冗余度检测和网络存储节省量上达到最优,基于 CDC 的检测技术次之,基于 FSP 的检测技术相对最差.而在额外存储开销方面(storage overhead),Sliding block 技术为提取数据所引入的元数据等额外存储开销最多,最高达 60%左右,基于 FSP 的检测技术次之,最多引入 50%的额外存储开销,基于 CDC 的检测技术效果最佳,引入的存储开销相对最低.

综上所述,目前还没有一种方法可以在所有数据集上都产生较好的结果,同时每种技术所需要的额外处理和存储空间以及具体数据集的使用模式和典型的负载,都对检测技术的选择起着至关重要的影响.

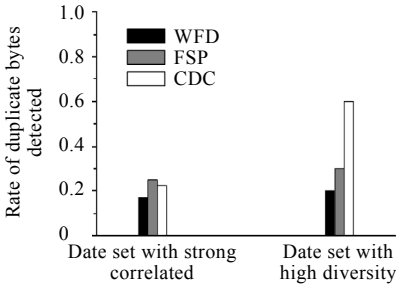
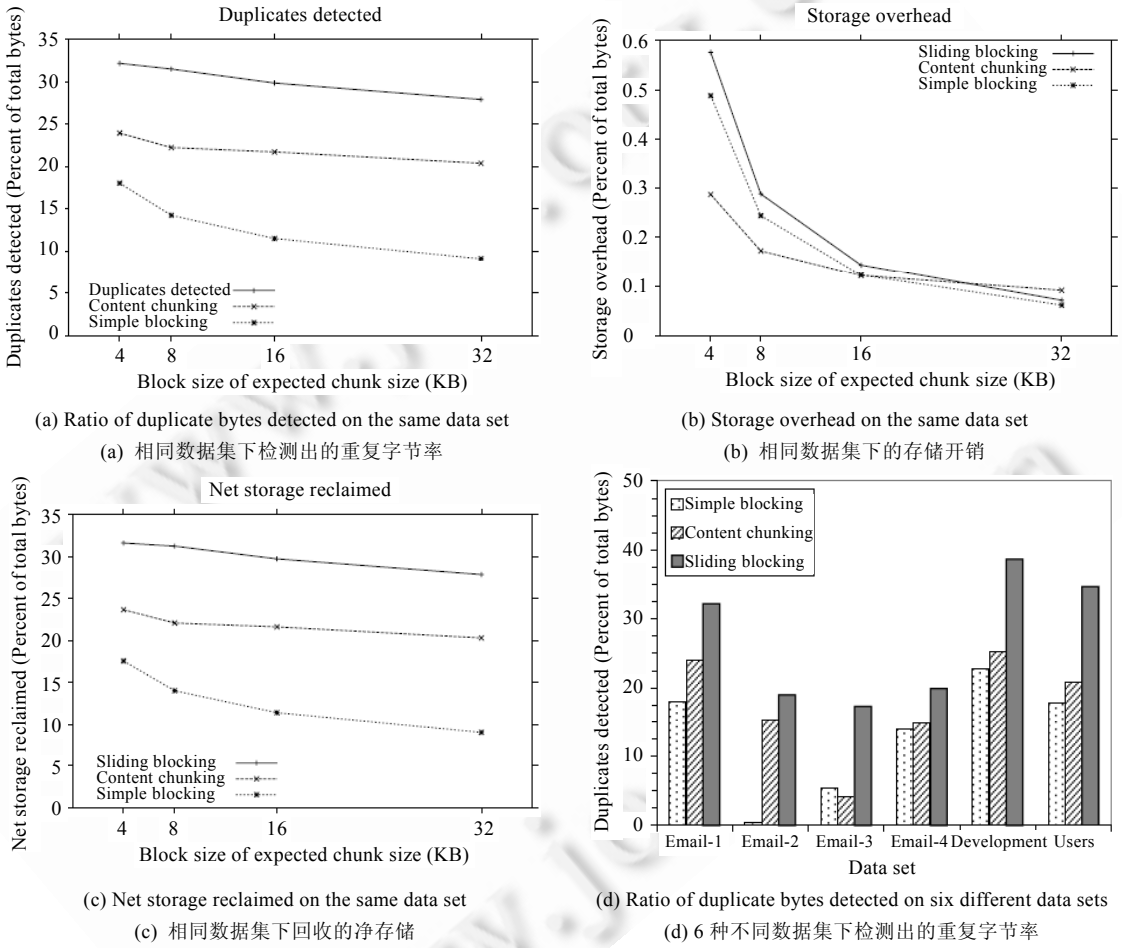


Fig.5 Performance comparison of WFD technique and identical data chunk detection techniques based on FSP algorithm and CDC algorithm[5]

图 5 WFD 技术与基于 FSP 算法和基于 CDC 算法的相同数据块检测技术的性能比较[5]



(a) Ratio of duplicate bytes detected on the same data set

(a) 相同数据集下检测出的重复字节率

(b) Storage overhead on the same data set

(b) 相同数据集下的存储开销

(c) Net storage reclaimed on the same data set

(c) 相同数据集下回收的净存储

(d) Ratio of duplicate bytes detected on six different data sets

(d) 6 种不同数据集下检测出的重复字节率

Fig.6 Performance comparison of identical data chunk detection techniques based on FSP algorithm, CDC algorithm, and sliding blocking method[7]

图 6 基于 FSP 算法、CDC 算法和滑动分块方法的相同数据块检测技术的性能比较[7]

为此,可得出如下结论:(1) 目前,没有一种方法可以作为通用的检测技术,因此,如何找到一个在时间开销、空间开销和匹配精度上都最佳的技术将是一个研究难点;(2) 如何在融合各技术特征的同时,对数据特性进行充分的分析和挖掘,找到其规律性来为系统开销的缩减提供一种技术支持,也将是一个值得研究的问题.

2 相似数据检测和编码技术***

从上一节的讨论中可以看出,因为数据自身相似性的特点,除了删除完全相同的数据可以缩减空间以外,研究发现相似数据的检测和编码处理也可以大幅度地减少存储空间,提高各种资源的利用率[10,11].

2.1 相似数据检测技术

2.1.1 基于 shingle 的检测技术

在相似性检测搜索中,理论上可以用像 unix diff 这样的程序把查询文档和系统中所有文档进行逐一比较[16],找到所有与其相似的文档.但这种做法效率很低.为此,研究者们提出为每个文档提取一组特征,这样,文档相似性问题就简化为集合相似性问题,如基于 shingle 的相似数据检测技术.

目前,关于数据集合相似性的定义有多种表示形式,文献[9]的定义如下:

一个数据段 D(可以是文件、数据块等)中一个连续的子序列称为一个 shingle,D 中所有大小为 w 的 shingle 集合定义为 $S(D,w)$.数据 A 和 B 的相似性定义为

$$r_w(A,B)=\frac{|S(A,w)\cap S(B,w)|}{|S(A,w)\cup S(B,w)|} \tag{1}$$

包含性定义为

$$c_w(A,B)=\frac{|S(A,w)\cap S(B,w)|}{|S(A,w)|},\ A\subseteq B \tag{2}$$

由公式(1)、(2)可以看出,数据相似性 $r_w(A,B)$与 w 的取值和文件的大小有关.设数据的大小为 B,则 $S(D,w)$中包含的 shingle 的个数为 $B-w+1$,当 B 较大时,单纯地对数据中所有 shingle 进行相似性处理使得系统的开销较大[8],因此研究者又提出了 3 种对 shingle 进行取样的技术:Min-Wise[9,19],Modm[6]和 Mins[12].Min-Wise 技术是通过将 shingle 的长度 w 和整数值进行映射产生随机哈希的公共集,在此相同的模式下进行随机最小独立置换的采样,从而得到采样集合.Modm 技术是通过在与 Min-Wise 同样的公共映射集中选择所有模 m 为 0 的哈希值对应的 shingle 组成取样集合.Mins 技术同样也是先将 shingle 和整数集进行映射,然后从中选择最小 s 个元素组成取样集合.

基于 shingle 的检测技术已应用到 AltaVista 搜索引擎[8]、基于群的大规模文件压缩技术[19]、模糊文件块匹配研究[12]和 REBL(redundancy elimination at the block level)[11]的冗余消除技术中.为了将系统开销最小化,REBL 在 Min-Wise 的基础上做了进一步的优化取样.

通过大量的实际应用,研究者们发现 shingle 及 Min-Wise,Modm 和 Mins 技术简单易实现,且适用性广,但仍然存在一定的缺陷[6]:(1) shingling(Min-Wise,Modm,Mins)用两个特征集的交集计算文件的相似性和包含性,计算开销很高.(2) 特征集的大小(shingle 数)决定了检测相似数据的精度,其值又取决于 shingling 的取样技术,而取样自身的误差使得结果可能出现较大的偏差.

2.1.2 基于 bloom filter 的检测技术

Bloom filter[14]是一种集合的表示方法,能够支持成员查询(查询某元素是否在该集合中).将 bloom filter 技术引入到相似数据检测中,可以弥补 shingle 中应用特征集交集计算文件相似性所导致的高计算开销,在性能与相似性匹配精度之间取得平衡.

在副本同步消除冗余的技术[6]中,系统应用 bloom filters 进行文件相似性检测,其中每个文件被看作是 bloom filter 中定义的一个集合,它的元素是文件按照 CDC 算法所划分出块的指纹值.有相同指纹值集合的文件由相同的 bloom filter 表示.相应地,相似的文件在它们的 bloom filter 中有大量共同的 1.因为 bloom filter 存在一个有限的错误匹配概率的特性,文献[6]的作者证明了在他们所提出的 CDC 与 bloom filter 相结合的技术中,错误匹配的位的部分取决于 bloom filter 的大小.

*** 本文中,编码技术是指相似数据检测后最终存储的技术.与通常所说的哈夫曼编码不同,该技术主要是用差分技术进行编码.

大量数据集测试表明,bloom filter 应用于相似性检测时,有以下几个优势:(1) bloom filter 对远程副本(存储和传输)系统来说很具有吸引力,这些系统中元数据开销通常很小.(2) bloom filter 使得快速匹配成为可能,因为匹配是按位与操作.(3) 因为 bloom filter 是一个集合的完整表示,而不是一个确定性样本(如 shingling),所以它们可以有效地确定包含关系,如 tar 文件和库.(4) 因为 bloom filter 有较低的元数据开销,所以可以进一步与滑动块技术或 CDC 技术结合以减少匹配空间.但是,在众多优点之下,bloom filter 也存在着一个问题:在相似性检测中,组成 bloom filter 的元素集合是文件特征提取后的集合,而这些元素间的全排列顺序没有在 bloom filter 的考虑范围之内,这可能导致相似性的判断与真实情况有所偏离.

2.1.3 基于模式匹配的检测技术

在相似数据检测技术中,基于 shingle 及其变种的技术和 bloom filter 成为发展的主流,但是也有部分学者研究模式匹配技术挖掘数据的特征,进行相似数据检测.

当前,模式匹配技术的研究主要集中在两个方面:单模式匹配和多模式匹配[17].单模式匹配算法和多模式匹配算法都是利用一定数量的公共子串进行两个文件间的相似性查找和判别.多模式匹配算法已应用在大型文件系统中进行相似数据的查找[16].

综合以上 3 种数据相似性检测技术的研究可知,bloom filter 技术在系统时间开销、空间开销和匹配精度上比 shingle 技术略占优势,而模式匹配技术需进行文件的整体扫描.同时,数据相似性检测技术是删除重复数据的重要组成部分,充分了解数据相似性匹配的规律,快速而准确地找到相似的数据块是决定重复数据删除效果的重要因素.因此,从存储系统性能的角度而言,亟待展开相似性检测技术的适用性和系统开销的实际研究.

2.2 相似数据编码技术

研究表明,在已有相似性检测技术的基础上,对具有较大相似度的数据进行编码处理,可以为整体系统节省大量的存储空间[18].当前,相似数据编码技术主要为 Delta 编码,它是通过用一个文件给另一个文件编码的方式进行数据的压缩.这种编码压缩适用的范围非常广,其中包括存储数据的多个版本、显示差别、合并变化、发布更新、存储备份、转换视频流等领域.

2.2.1 基于 diff 的 delta 编码技术

Unix 的 diff[34,35]是生成 delta 编码的经典算法.它在版本控制和配置管理系统中使用最为广泛.Diff 算法的主要思想是通过考虑整行[34]来找到最长公共子串的估计值.它并不检查字节位置的所有可能的组合,因此比计算字节级别的最长公共子串(longest common subsequence,简称 LCS)要快很多.对两个字符串,LCS 的含义[18]是同时包含在这两个字符串里的一个最长字符序列,LCS 的长度是这两个字符串相似性的很好的度量.

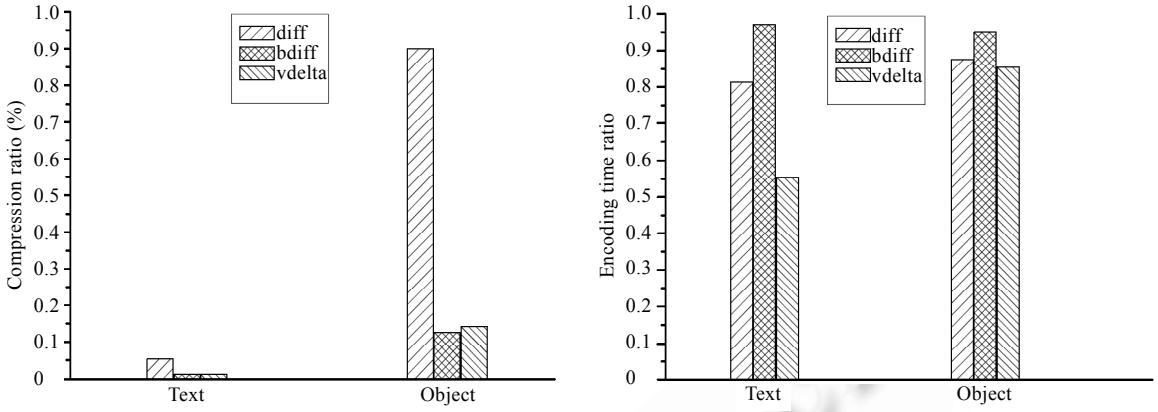
版本控制系统 SCCS(source code control system)和 RCS(revision control system)[36,37]都应用 diff 算法来存储和显示差别,RCS 还用它进行合并.通过存储相对于一个基本版本的 delta 值,这些系统与存储每个版本的整体的系统相比节省了大量的磁盘空间.

但是 diff 算法有其局限性.它仅适用于文本文件的差异查找编码,并且其粒度仅限于行查找(diff 算法只能找到共同的行).当对二进制代码进行版本控制,而不仅仅是源文件文本时,diff 算法必须先把二进制代码映射为文本,然后再应用 diff.

2.2.2 基于 bdiff 的 delta 编码技术

如今需要管理的数据有很多都是二进制源文件格式[18],例如 Word 处理器文件,电子数据表数据,电子和机械 CAD 数据,声音和图像,然而,基于 diff 算法的编码技术只能处理文本文件,因此研究者们针对这个问题提出了一种新的算法 bdiff[18].Bdiff 是一种压缩算法,它与 diff 一样,应用了 LCS 度量方法,但它对任何字节流类型都适用,是 Tichy 提出的块移动算法的修改版本[38].

文献[18]通过实验对 diff,bdiff 和 vdelta 的性能进行了比较,如图 7 所示.从图 7 可以看出,bdiff 算法在文本文件和目标文件的压缩率上均优于 diff 算法[18],但压缩速率比 diff 算法要慢.



(a) Resultant compression ratio for GNU GCC text files and object files

(a) 对 GNU GCC 文本文件和对象文件的总压缩率

(b) Relative encoding time ratio for GNU GCC text files and object files

(b) 对 GNU GCC 文本文件和对象文件的相对编码速率

Fig.7 Performance comparison of diff, bdiff and vdelta[18]

图 7 Diff,bdiff 和 vdelta 的性能比较[18]

2.2.3 基于 vdelta 的 delta 编码技术

Vdelta 是一种结合数据压缩和数据差异的新技术,同样也应用了 LCS 度量方法,对任何字节流类型都适用. Vdelta 也是 Tichy 提出的块移动算法的改进[38].它借鉴 1978 年 Ziv-Lempel 的压缩技术的数据解析方法,使用哈希表方式建立索引,而不是 bdiff 中的后缀树方式.

经大量测试分析,Vdelta 和 bdiff 在压缩率上都优于 diff 算法[18],如图 7 所示.对文本文件,bdiff 和 vdelta 压缩率较为接近,而对目标代码 bdiff 的压缩效果最好.在压缩速率上,vdelta 压缩时间最少,速度最快.

2.3 小 结

鉴于文件之间的相似性原理[9],利用以上各种检测和编码技术可以使数据占用空间大幅度缩减,但每种技术都有其各自的特点和不足.相似性检测技术存在着如何平衡检测精度和系统开销的问题,而相似性编码技术存在着编码效率和适用范围的问题.

分析目前的研究现状,可以得出如下结论:(1) 目前,相似性检测的 3 种技术在系统时间开销、空间开销和匹配精度上各有优势,但是没有一种技术同时具有这 3 个方面的特点,如何融合这 3 种技术将是一个研究的难点;(2) 因相似性检测技术在自身设计上存在局限性,如何通过结合统计学和数据挖掘领域的各种技术(聚集、维规约、特征子集选择、特征创建等)以及综合考虑简单匹配系数相似、Jaccard 系数相似、余弦相似等多种相似度来打破当前各技术的局限,还需要进一步研究;(3) 在相似性编码技术中,新的压缩理论与技术和更有效的数学模型不断出现,如何通过引进压缩算法开发新的技术,更有效地优化储存空间的使用,还需要进一步研究.

3 重复数据删除技术的可靠性研究

重复数据删除技术的应用,使文件最终以共享数据对象或数据块的引用集合存储.尽管该项技术与传统的跨文件压缩方法相比有很高的压缩率,但它降低了存储系统的可靠性,因为几个关键数据块的丢失或错误可能导致许多文件的丢失和错误发生.为了弥补这一缺陷,一些研究者对其可靠性进行了深入的研究[7,12,20,23].

3.1 冗余复制技术

一些研究者提出通过消除冗余所节省空间的一部分复制一些块[7,20,23],策略性地增加冗余来提高系统可靠性.鉴于单纯为所有数据增加冗余所导致的大量空间消耗,研究者们提出了不同的处理方法:

(1) 提出根据块的重要度来决定块的重复度[7,20,23],它按照基于块权重的复制方法来增加数据冗余.在该技

术中设定即使一个块只有一个依赖关系,也必须被保护.所以其启发式规则为每个块保存至少两个复制.文献[20]提出的该可靠性技术模型不会影响重复数据删除技术带来的存储空间的节约,它利用合适的启发式规则和参数变化,占用大约一半的存储空间,取得了比传统跨文件压缩方法更高的可靠性.

(2) 针对引用数据进行冗余复制,即对实际有用的数据,而非传统的所有数据[7].研究发现,引用数据或内容固定的数据占据了日常数据量的大部分,并且增长迅速[7].而引用数据的特征是存在大量的相似数据,且不能被改变,并且需要保持很长的时间.因此文献[7]提出了一个针对引用数据实现的可靠性存储系统.该系统通过管理数据的唯一块可靠、高效地存储大量的相似数据,并允许被选择的数据被高效删除.其中系统可靠性的设计是基于块与块之间的信息内容来管理数据存储的.

(3) 在不同应用或层次上增加冗余数据提高系统的可靠性.RAID[39]是通过引进冗余并以此保证存储系统的可靠性.OceanStore[40]对持久数据提高全局的连续访问,并使用自动复制策略来提高系统可靠性.FARSITE[41]是一个分布式系统,它通过复制文件系统的元数据来提供可靠性.

3.2 纠删码技术

在存储系统中,单纯地通过增加完全副本冗余来提高系统的可靠性并不能保证在错误出现时,数据仍具有持久性和可靠性.因此一些研究者提出利用纠删码技术对数据做一定的冗余来增加系统的可靠性[12].

纠删码技术是指将要存储的数据切分为 k 个部分,然后通过编码算法变换为 $n(n>k)$个部分,其中任意 $k'(k'\geq k)$个部分可以用来恢复原始数据[42].当 $k'=k$ 时,称编码算法具有最大距离分割性质(maximum distance separable,简称 MDS).

纠删码主要分为 4 大类:Reed Solomon Codes,Parity Array Codes,Parity-check Codes 和 LDPC(low density parity check) Codes.其中,Reed Solomon Codes 是基于有限域运算的,后三者主要是基于 XOR 运算.

每种纠删码(erasure codes)都有其各自的优缺点:(1) Reed Solomon 码:① MDS 码,具有最优的存储利用率;② 容错能力 k'可以为任意数;③ 数据出度总是大于 k',从而有较高的更新(update)复杂度;④ 复杂的代数运算.(2) Parity Array 码:主要是通过单元的二维空间几何分布构成的,适用于 RAID 系统.(3) Parity-check 码:从奇偶校验码发展而来的多维码,特点是简单且完全基于 XOR 运算,容易实现,且更新(update)和解码(decode)等操作都具有较好的局部性,其缺点是非 MDS,主要适合于大规模存储系统.(4) LDPC 码:基于 Tanner 图发展起来的一维码,其构造主要是基于图论和蒙特卡洛法.其优点是可以完全用 XOR 运算实现,并且其存储利用率接近 MDS.其解码算法主要是采用迭代方法实现,可以达到很高的容错能力,但是容错能力越高,其构造越不规则,从而导致不易实现.

目前在重复数据删除系统中应用纠删码技术来提高系统可靠性的研究还较为简单,仅局限于 Reed Solomon 纠删码,因此,如何引入其他纠删码技术提高系统可靠性,将有待进一步的研究.

3.3 小 结

通过上述论述可知,提高系统可靠性的两种主要技术本质上都是基于增加冗余数据,但又有其各自的适用范围和技术特点:(1) 冗余复制技术主要是完全副本拷贝,在进行相同数据块检测和删除时,应用该技术适度地加入冗余,可在保证节省磁盘空间和网络带宽的前提下,增加系统的可靠性.(2) 纠删码技术主要是通过差分控制编码,进行数据的检错和纠错,相对于完全副本方式,纠删码带来了一定的计算量,也增加了系统设计和实现的复杂度.但在一定情况下,它也能在增加较少冗余数据的情况下提供与完全副本方式相同的可靠性.因此,如何挖掘数据的各种特性,针对不同的数据类型,设计有效的度量方式,适度地增加副本或校验码以高效地提高系统的可靠性,是一个需要深入研究的问题.(3) 因重复数据删除技术固有的风险——硬件故障引起的灾难性数据丢失,单纯地利用增加冗余数据的方法效果不佳,因此需要引入其他机制,如 RAID-5 奇偶校验、不同数据替代、错误检测与恢复等技术,同时结合硬件和故障统计数据,如磁盘的故障间平均时间,解决数据分布或块存储问题,以提高系统的可靠性.

4 重复数据删除技术的性能研究

数据的检测对比等过程都需要消耗大量的系统资源,严重影响存储系统中数据访问的性能.因此,为了提高系统性能,研究者们提出了一些解决方案.

4.1 减轻磁盘瓶颈技术

在重复数据删除系统中,为了节约成本,一些系统仅有少量的内存,因而不能支持所有的数据索引一次性地进入内存进行检测,从而导致了大量的磁盘访问.针对这种情况,文献[21]中描述了 Data Domain 重复数据删除文件系统中用于减轻磁盘瓶颈的 3 种技术:摘要向量、基于流的块排列和局部性保持.应用这 3 种技术,实现了一个高吞吐率、低开销的相同块删除存储系统.

(1) 摘要向量技术

摘要向量技术用于减少在磁盘中查找不存在块的次数.摘要向量可以看作是一种处于内存中,对块索引的摘要.如果摘要向量指出一个块不在索引内,则不需要进一步查找.如果摘要向量指出这个块在块索引中,那么很有可能这个块就在块索引内,但并不是一定的,因为摘要向量利用 Bloom filter 来实现,一个 Bloom filter 用一个 m 位向量来概括在块索引中 n 个块指纹值的存在信息,且 Bloom filter 的特点是允许错误肯定(认为一个不在集合中的元素在集合中)的存在.

因为摘要向量是在内存中的数据结构,所以系统关机后,会将其写到 disk 上,启动后,又会从 disk 上读入内存中.为了处理停电和不正常的关机情况,系统会周期性地将摘要向量备份到磁盘上,并设置检查点.当恢复时,系统会载入最近备份的副本,并处理从最近一个检查点添加到系统中的数据,将其信息加入到摘要向量中.

(2) 基于流的块排列技术

基于流的块排列技术(stream-informed segment layout,简称 SISL)为块数据和块描述符提供了更好的空间局部性,并且使局部 cache 缓存成为可能.SISL 主要特点是针对同一个数据流,使得新数据块被存放在一起,块描述符也被存放在一起.SISL 能够带来以下益处:① 当同一个数据流的多个数据块被写入到同一个容器中时,在进行读取操作重建这个数据流时,可以大幅减少磁盘 I/O 次数,使系统达到较高的读性能.② 在同一数据流中,相邻新数据块的块描述符和压缩数据分别被线性地装入相同容器的元数据段和数据段.这种装入方法为其后相似的数据流提供了一种访问局部性,使得 cache 的局部缓存工作更有效.③ 元数据段和数据段分开存储,而且元数据段比数据段要小得多.

(3) 局部性保持技术

系统利用局部性保持技术(locality preserved caching,简称 LPC)来加速辨认重复块的过程.针对重复数据检测,传统的 Cache 并不适用于缓存指纹值、哈希值或者描述符,因为指纹值在本质上是随机的.因此,如果没有进行完全的索引访问,预言下一个块的索引位置是非常困难的,从而也导致了传统 cache 的命中率是非常低的.LPC 的应用使得块重复局部性提高,如果一个块是重复的,那么附近的块很可能已经被缓存了.

实验结果表明,在重复数据删除的现实环境中,把以上 3 种技术结合在一起,可以省去 99%的磁盘访问.这些技术对于一个双核计算机系统而言,就是利用其 90%的 CPU 性能和一个 15 个磁盘的磁盘柜,即可达到 100MB/s 的单工流吞吐率和 210MB/s 的双工流吞吐率[30].

4.2 提高数据搜索速度的技术

重复数据的检测比对过程给系统带来了巨大的时间开销.当数据的所有特征存在于 1 个索引中时会导致系统的可扩展性差,同时导致特征值匹配的时间开销过大,因此一些研究者提出了提高数据搜索匹配速度的关键技术:把索引分片为多个小索引,并存储在多个服务器上,增加搜索的可扩展性及可并行性[22].同时,当有新文档加入系统时,选择一部分分片服务器来存储它的特征;在重复数据搜索时,也只在一部分分片服务器上查询.

为了加速搜索的速度,文献[22]提出了一个新的索引分片和文档路由策略.它将一个文档的所有特征路由在一起,而并不把文档的每个特征独立地路由.在吸收文档时,文档路由策略用特征提取算法提取出文档的特征集,基于这些特征,选择一部分索引分片.文档被路由到这些分片上,并加入那里的索引中.在查询时,用同样的特

征提取和文档路由算法选择查询的分片,在所选择的分片上查询该文档的特征,并把结果进行归并.

实验结果表明,快速搜索技术解决了数据匹配中时间复杂度较高的问题,并且扩展性很好,对搜索结果的准确性和回取的影响很小.

4.3 提高相似数据编码速度的技术

在对大规模文件使用 delta 压缩时,一些研究者发现 Delta 压缩技术虽然能够有效地减少数据的占用空间[19],但因 delta 压缩简洁地描述某一文件相对于另一文件的编码,使得数据最终压缩比与参考文件的选择存在密切的关系.因此,文献[19]研究了一种优化的 delta 编码技术,为基于群的 Delta 压缩提出一种新的框架.在这种框架下,使用文本群技术以削剪可能出现双 delta 编码的图,使得在带权有向图中计算有最大权重分支的编码效率较低的问题得到改善.

基于群的 Delta 压缩编码技术的基本思想是将完全图 G 修剪成稀疏子图 G',然后在这个子图中找到最佳 delta 编码方案.经实验测试,对集合的基于群的 delta 压缩与对每个文件单独使用 tar 和 gzip 进行压缩相比,在压缩率上有重大的改进.

4.4 小 结

本节介绍了重复数据删除技术的性能研究情况,各项技术都在不同程度上提高了系统的性能.(1) 减轻磁盘瓶颈的几种技术在 Data Domain 的重复数据删除文件系统下实现,减少了 98.94%的磁盘访问[21],且该技术具有可扩展性和通用性.(2) 提高数据搜索速度的技术主要是解决了数据匹配中时间复杂度较高的问题.它利用增加服务器的方法来解决.(3) 提高相似数据编码速度的技术对系统的时间和空间开销均有一定的改善,对该技术局限于每个目标文件相对于一个单独的参照文件进行编码压缩处理的情况,并且只将压缩和解压作为一个整体集合考虑,不允许单个文件加到集合中或者从集合中取回.

分析目前的研究现状,我们可以得出如下结论:(1) 目前,在重复数据删除系统中,提高系统性能的各关键技术均有一定的缺陷.如何借鉴已有技术的优点,设计高效通用的技术方案,将系统空间时间开销综合起来考虑,需要更深入的研究.(2) 在提高相似数据编码速度的技术中,因编码参考文件选择的重要性,如何将单一参考文件进行扩展,利用一些新的局部最小求解算法,如蚂蚁算法、拟物算法等,选择最佳的参考文件进行编码,大幅度提高压缩率,优化储存空间的使用也是一个值得研究的问题.

5 结束语

随着数字信息的爆炸式增长,重复数据的存在形式有很多,从文件级和数据块级完全相同的数据到字节级的相似数据都存在着可探究的空间.针对相同文件,研究者提出利用 hash 技术提取文件特征进行全文件数据的检测和匹配,然而由于比较粒度较粗,不能检测和消除实现更多的相同数据的冗余,于是,研究者又提出了基于块的相同数据检测技术(如,固定块大小、可变块大小和滑动块检测技术),但这些技术没有有效利用细粒度数据相似的特性,在某种程度上降低了重复数据匹配的性能,因此,一些研究者们又提出了字节级别的相似数据删除技术,其中包括 shingle、bloom filter、模式匹配检测技术和 delta 编码技术.与此同时,因重复数据删除技术导致的系统可靠性和性能问题,一些研究者又针对重复数据删除技术的可靠性和性能进行了研究.在此基础上,本文综合分析了目前重复数据删除各项技术研究的现状,探讨了重复数据删除的发展趋势.

从目前的研究来看,我们可以得出如下结论:(1) 如何挖掘不同类型的数据特性,快速而准确地检测到重复数据,同时具有较低的空间开销,仍然存在着可探究的空间.(2) 因数据相似性检测技术设计上存在一定的局限性,如何在融合各技术特征的同时,通过结合统计学和数据挖掘领域的各种技术,对数据特性进行充分的分析和挖掘,找到其规律性的认识来弥补重复数据删除技术上的不足,提高整体系统的性能,还需要进一步研究.(3) 因新的压缩理论与技术或更有效的数学模型不断出现,如何通过引进压缩算法开发新的技术或将已有技术结合在一起,有效地优化储存空间,还需要进一步研究.(4) 已有的可靠性技术是基于增加冗余数据的,其技术模型具有简单、高效的特点,但在存储开销和系统性能方面存在一定的局限性.因此,如何针对不同的数据类型,适度地

增加冗余数据来提高系统的可靠性或引入其他机制的可靠性设计,还需要进一步研究.(5) 在重复数据删除的过程中,额外增加的系统开销是不可忽略的,在实际应用过程中,常常需要在简单性和性能两方面做出折衷选择.因此,如何在融合各种现有技术的同时,提供通用性、可扩展性和自适应性,尽可能地减少重复数据检测和删除所带来的系统开销,还需要进一步研究.

References:

[1] McKnight J, Asaro T, Babineau B. Digital archiving: End-User survey and market forecast 2006-2010. 2006. http://www.enterprisestrategygroup.com/ESGPublications/ReportDetail.asp?ReportID=591

[2] Muthitacharoen A, Chen B, Maziéres D. A low-bandwidth network file system. In: Proc. of the 18th ACM Symp. on Operating System Principles (SOSP 2001). New York: ACM Press, 2001. 174−187.

[3] Bolosky WJ, Corbin S, Goebel D, Douceur JR. Single instance storage in Windows 2000. In: Proc. of the 4th Usenix Windows System Symp. Berkeley: USENIX Association, 2000. 13−24.

[4] Bobbarjung DR, Jagannathan S, Dubnicki C. Improving duplicate elimination in storage systems. ACM Trans. on Storage, 2006, 2(4):424−448.

[5] Policroniades C, Pratt I. Alternatives for detecting redundancy in storage systems data. In: Proc. of the 2004 USENIX Annual Technical Conf. (USENIX 2004). Berkeley: USENIX Association, 2004. 73−86.

[6] Jain N, Dahlin M, Tewari R. Taper: Tiered approach for eliminating redundancy in replica synchronization. In: Proc. of the 4th Usenix Conf. on File and Storage Technologies (FAST 2005). Berkeley: USENIX Association, 2005. 281−294.

[7] Denehy TE, Hsu WW. Duplicate management for reference data. IBM Research Report, RJ 10305 (A0310-017), IBM Research Division, 2003.

[8] Broder AZ. Identifying and filtering near-duplicate documents. In: Giancarlo R, Sankoff D, eds. Proc. of the 11th Annual Symp. on Combinatorial Pattern Matching. London: Springer-Verlag, 2000. 1−10.

[9] Broder AZ. On the resemblance and containment of documents. In: Carpentieri B, De Santis A, Vaccaro U, Storer JA, eds. Proc. of the Compression and Complexity of SEQUENCES 1997. Washington: IEEE Computer Society Press, 1997. 21−29.

[10] Douglis F, Iyengar A. Application-Specific delta encoding via resemblance detection. In: Proc. of the 2003 USENIX Annual Technical Conf. (USENIX 2003). Berkeley: USENIX Association, 2003. 113−126.

[11] Kulkarni P, Douglis F, Lavoie JD, Tracey JM. Redundancy elimination within large collections of files. In: Proc. of the 2004 Usenix Annual Technical Conf. (USENIX 2004). Berkeley: USENIX Association, 2004. 59−72.

[12] Han B, Keleher P. Implementation and performance evaluation of fuzzy file block matching. In: Proc. of the 2007 USENIX Annual Technical Conf. (USENIX 2007). Berkeley: USENIX Association, 2007. 199−204.

[13] Bloom BH. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970,13(7):422−426. [doi: 10.1145/362686.362692]

[14] Broder AZ, Mitzenmacher M. Network applications of bloom filters: A survey. Internet Mathematics, 2003,1(4):485−509.

[15] Mitzenmacher M. Compressed bloom filters. IEEE/ACM Trans. on Networking (TON), 2002,10(5):604–612. [doi: 10.1109/TNET.2002.803864]

[16] Manber U. Finding similar files in a large file system. In: Proc. of the USENIX Winter 1994 Technical Conf. Berkeley: USENIX Association, 1994. 1–10.

[17] Wu S, Manber U. Agrep—A fast approximate pattern-matching tool. In: Proc. of the USENIX Winter 1992 Technical Conf. Berkeley: USENIX Association, 1992. 153–162.

[18] Hunt JJ, Vo KP, Tichy WF. Delta algorithms an empiriecal analysis. ACM Trans. on Software Engineering and Methodology, 1998, 7(4):192–214. [doi: 10.1145/279310.279321]

[19] Ouyang Z, Memon N, Suel T, Trendafilov D. Cluster-Based delta compression of a collection of files. In: Proc. of the 3rd Int'l Conf. on Web Information Systems Engineering. Washington: IEEE Computer Society Press, 2006. 257–266.

[20] Bhagwat D, Pollack K, Long DDE, Schwarz T, Miller EL, Pâris JF. Providing high reliability in a minimum redundancy archival storage system. In: Proc. of the 14th Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006). Washington: IEEE Computer Society Press, 2006. 413–421.

[21] Zhu B, Li K. Avoiding the disk bottleneck in the data domain deduplication file system. In: Proc. of the 6th Usenix Conf. on File and Storage Technologies (FAST 2008). Berkeley: USENIX Association, 2008. 269–282.

[22] Bhagwat D, Eshghi K, Mehra P. Content-Based document routing and index partitioning for scalable similarity-based searches in a large corpus. In: Berkhin P, Caruana R, Wu XD, Gaffney S, eds. Proc. of the 13th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD 2007). New York: ACM Press, 2007. 105–112.

[23] You LL, Pollack KT, Long DDE. Deep store: An archival storage system architecture. In: Proc. of the 21st Int'l Conf. on Data Engineering (ICDE 2005). Washington: IEEE Computer Society Press, 2005. 804–815.

[24] Quinlan S, Dorward S. Venti: A new approach to archival storage. In: Proc. of the 1st Usenix Conf. on File and Storage Technologies (FAST 2002). Berkeley: USENIX Association, 2002. 89–102.

[25] Sapuntzakis CP, Chandra R, Pfaff B, Chow J, Lam MS, Rosenblum M. Optimizing the migration of virtual computers. In: Proc. of the 5th Symp. on Operating Systems Design and Implementation (OSDI 2002). New York: ACM Press, 2002. 377–390.

[26] Rabin MO. Fingerprinting by random polynomials. Technical Report, CRCT TR-15-81, Harvard University, 1981.

[27] Rivest R. The MD5 message-digest algorithm. 1992. http://www.python.org/doc/current/lib/module-md5.html

[28] U.S. National Institute of Standards and Technology (NIST). Federal Information Processing Standards (FIPS) Publication 180-1: Secure Hash Standard. 1995. http://www.itl.nist.gov/fipspubs/fip180-1.htm

[29] U.S. National Institute of Standards and Technology (NIST). Federal Information Processing Standards (FIPS) Publication 180-2: Secure Hash Standard. 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[30] Moreton TD, Pratt IA, Harris TL. Storage, mutability and naming in Pasta. In: Proc. of the Int'l Workshop on Peer-to-Peer Computing at Networking 2002 Workshops. Berlin, Heidelberg: Springer-Verlag, 2002. 215–219.

[31] Cox LP, Murray CD, Noble BD. Pastiche: Making backup cheap and easy. In: Proc. of the 5th Symp. on Operating Systems Design and Implementation (OSDI 2002). Berkeley: USENIX Association, 2002. 285–298.

[32] Rhea SC, Liang K, Brewer E. Value-Based Web caching. In: Proc. of the 12th Int'l Conf. on World Wide Web (WWW 2003). New York: ACM Press, 2003. 619–628.

[33] Tridgell A. Rolling checksum. 1998. http://rsync.samba.org./tech_report/node3.html

[34] Hunt JW, McIllroy MC. An algorithm for differential file comparison. Technical Report, No.41, Computing Science, AT&T Bell Laboratories, 1976.

[35] Hunt JW, Szymanske TG. A fast algorithm for computing longest common subsequences. Communications of the ACM, 1977, 20(5):350–353. [doi: 10.1145/359581.359603]

[36] Rochkind MJ. The source code control system. IEEE Trans. on Software Engineering, 1975,SE-1(4):364–370.

[37] Tichy WF. RCS—A system for version control. Software-Practice and Experience, 1985,15(7):637–654. [doi: 10.1002/spe.4380150703]

[38] Tichy WF. The string-to-string correction problem with block moves. ACM Trans. on Computer Systems, 1984,2(4):309–321. [doi: 10.1145/357401.357404]

[39] Chen PM, Lee EK, Gibson GA, Katz RH, Patterson DA. RAID: High-Performance, reliable secondary storage. ACM Computing Surveys, 1994,26(2):145–185. [doi: 10.1145/176979.176981]

[40] Kubiatowicz J, Bindel D, Chen Y, Czerwinski S, Eaton P, Geels D, Gummadi R, Rhea S, Weatherspoon H, Weimer W, Wells C, Zhao B. Oceanstore: An architecture for global-scale persistent storage. In: Proc. of the 9th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000). New York: ACM Press, 2000. 190–201.

[41] Adya A, Bolosky WJ, Castro M, Chaiken R, Cermak G, Douceur JR, Howell J, Lorch JR, Theimer M, Wattenhofer R. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In: Proc. of the 5th Symp. on Operating Systems Design and Implementation (OSDI 2002). New York: ACM Press, 2002. 1–14.

[42] Rizzo L. Effective erasure codes for reliable computer communication protocols. ACM SIGCOMM Computer Communication Review, 1997,27(2):24–36. [doi: 10.1145/263876.263881]

敖莉(1983－),女,内蒙古通辽人,硕士,主要研究领域为网络存储技术.

李明强(1984－),男,博士生,主要研究领域为存储可靠性,网络存储.

舒继武(1968－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络存储,存储安全,并行处理技术.