基于改进Sparse Indexing的多负载消冗方法

王 灿[1,2]，秦志光[1,2]，杨 磊[1,2]，杨 皓[1,2]

(1. 电子科技大学计算机科学与工程学院 成都 611731; 2. 网络与数据安全四川省重点实验室 成都 611731)

【摘要】针对现有的Sparse Indexing方法不能有效处理小文件备份负载的问题，提出了一种以Broder扩展定理为理论依据的最小特征采样算法，该算法可以对不同形式的备份负载进行有效的特征采样。在此算法的基础上，设计了一种多负载重复数据消除方法，该方法通过对备份负载进行特征采样，仅在内存中维护完整索引的一个很小的子集，并通过批量读入分块标识符，摊销了磁盘访问开销，提高了吞吐量。实验结果表明，该方法对混合备份负载的压缩比是Sparse Indexing的2.04倍，而吞吐量与Sparse Indexing相当。该方法适用于需要处理多种形式备份负载的高性能重复数据消除系统。

关 键 词 重复数据消除; 磁盘瓶颈; 最小特征采样; 稀疏索引; 吞吐量

中图分类号 TP309.3 文献标志码 A doi:10.3969/j.issn.1001-0548.2013.05.017

Multiple-Loads Deduplication Method Based on Improved Sparse Indexing

WANG Can[1,2], QIN Zhi-guang[1,2], YANG Lei[1,2], and YANG Hao[1,2]

(1. School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 611731;
2. Network and Data Security Key Laboratory of Sichuan Province Chengdu 611731)

Abstract To address the problem that the sparse indexing can not deduplicate the backup load based on small files effectively, a min-feature sampling algorithm based on the Broder's extension theorem is proposed. In addition, a deduplication method for multiple backup loads, which is on the basis of the min-feature sampling algorithm, is presented. This method only maintains a very small subset of the full index in the RAM by sampling the backup load, and the cost of disk accesses is amortized by loading the chunk IDs in batches. As a result, the throughput of the method is improved effectively. The experimental results indicate that the compression ratio of the method on the mixed backup loads is 2.04 times of the sparse indexing, and its throughput is almost equal to the sparse indexing. This method is applicable to the high-performance deduplication systems which need to process backup loads of multiple types.

Key words deduplication; disk bottleneck; minimum feature sampling; sparse index; throughput

近年来，随着各种新兴网络应用的飞速发展，电子数据量增长迅猛：2007年新增的电子数据总量为281 EB，预计2011年该数字会增长10倍以上[1]。如何有效减小海量数据存储和备份的空间开销是一个重要的问题。重复数据消除技术[2-5]是一种重要的存储优化技术，该技术利用数据对象之间的信息冗余，可以获得远高于传统压缩方法及增量备份方法的空间利用率。该技术将数据对象划分为连续的、互不交叠的分块(Chunk)，计算每个分块的哈希值作为其唯一标识符(ChunkID)，并将其存入到分块索引表(HT_{ID})中。数据对象被表示成各分块标识符按顺序连接在一起的哈希链，并以分块为单位进行存储。每当写入新数据对象时，用该数据对象的各分块标识符在索引表中进行分块存在性查询(CEQ)，并只存储那些新的分块及其标识符。由于重复的数据分块不再多次存储和传送，因此可以大幅度减小存储空间占用量和远程传送时的网络带宽占用。

传统的重复数据消除系统中，所有非重复的Chunk都对应HT_{ID}中一条记录，这种结构称为平面型的索引结构。随着数据规模的增大，平面型HT_{ID}的大小迅速增加，不能全部装入内存，需要分页到磁盘上。由于哈希函数的特性，ChunkID是近似随机均匀分布的，因此传统方法无法有效预取和缓存，这就导致几乎每次CEQ都会引发磁盘访问，严重降低了吞吐量。例如，当Chunk的平均大小为4 KB时，如果每次磁盘访问时间需要4 ms，则系统平均每秒

收稿日期：2011 – 10 – 12; 修回日期：2012 – 02 –21
基金项目：国家科技重大专项(2011ZX03002-002-03)；教育部培育基金(708078)；国家自然科学基金(60873075, 60973118)
作者简介：王灿(1977 –)，男，博士，主要从事信息安全、存储优化技术等方面的研究.

钟只能处理250个Chunk，吞吐量仅为1 MB/S。

Sparse Indexing[6]是一种解决上述磁盘瓶颈问题的方法，该方法利用了数据的局部性，以分段为单位成批地对分块进行重复数据消除，摊销了磁盘访问开销，对具有显著局部性的传统备份负载，性能较优。但Sparse Indexing在处理缺乏局部性的非传统备份负载时，重复数据消除性能很差。

针对上述问题，本文提出一种基于改进Sparse Indexing的多负载重复数据消除方法(简写为MDM)，该方法能够有效处理传统备份负载、非传统备份负载以及它们的混合负载，从而实现跨负载的重复数据消除。

1 Sparse Indexing方法

Sparse Indexing利用了传统备份负载的局部性来提高系统吞吐量，下面先对数据局部性的含义和Sparse Indexing的主要过程及其缺点进行说明。

1.1 数据的局部性

传统的备份负载通常是每天(/每周)将包含很多文件的目录树链接为较大的备份映像文件，因此传统备份负载是由数据流组成的，其特点是组成备份负载的数据对象较大，通常为GB级。由于同一目录下的文件在邻近的备份版本之间变动很小，且几乎是以相同的顺序出现的，因此邻近的备份数据流之间有大段的重复数据。换而言之，如果本次备份数据流中*A*、*B*、*C* 3个文件的Chunk是按照该顺序出现的，那么在下一次备份数据流中如果*A*的Chunk出现了，随之出现*B*和*C*的Chunk的概率非常高。传统备份负载的这一特性称为数据的局部性[6-7]。

1.2 Sparse Indexing的主要过程

Sparse Indexing处理备份负载的主要过程如下[6]：

1) 对备份负载进行预处理。

① 用基于内容的分块算法TTTD[8]将备份负载划分为互不交叠的连续Chunk。

② 用分段算法[6]将步骤①中得到的Chunk序列划分为互不交叠的连续分段(Segment)，Segment的大小期望值为10 MB。每个Segment在系统中用一个分段清单(Manifest)表示，该数据结构中记录了该Segment所包含全部Chunk的ChunkID、大小、存储地址以及在Segment中的排列顺序等元数据信息。

2) 从已存储分段中选择相似分段。

① 对每个新到的Segment(记为Sgt_N)进行特征采样，取出其中那些满足预设采样条件的ChunkID(选中的ChunkID称为Hook)。

② 用步骤①选出的Hook查询稀疏索引(sparse index，SPI)，得到所有包含这些Hook的Manifest集合。这里的稀疏索引是一个从Hook到包含该Hook的Manifest的映射表，SPI常驻内存。

③ 从步骤②得到的Manifest集合中选出一定数量的与Sgt_N共享的Hook数量最多的Manifest，被选中的Manifest称为Champion。

3) 基于相似分段进行重复数据消除。

① 从磁盘上将Champion调入内存作为检索源，对Sgt_N进行重复数据消除。

② 为Sgt_N创建对应的分段清单mf_N，并将mf_N存储到磁盘上。

③ 用Sgt_N的Hook更新SPI：对已有的Hook，将mf_N的标识符mid_N添加到该Hook指向的ManifestID链表中；对新的Hook，新增一条以该Hook为主键的记录，并将mid_N添加到该记录的ManifestID链表中。

1.3 Sparse Indexing特征采样算法的缺点

如前所述，Sparse Indexing能够有效处理基于数据流的传统备份负载，但随着对服务精细化和即时性要求的提高，在很多应用环境中，备份负载不再是由大的数据流组成，而是由细粒度的文件组成，这些较小的数据对象来自于分散的源，并且到达的顺序完全是随机的。例如多个用户随机提交的以文件为单位的备份和恢复请求[9]；连续数据保护(CDP)[10]，任何文件一旦发生改变，就立即要备份。这样，在一段时间内到达的文件之间局部性很弱。

Sparse Indexing在处理上述非传统备份负载时，由于分段是不跨文件的，再加上大部分文件的大小都小于Segment的期望值(10 MB)，因此得到的分段结果中大部分Segment就是由一个文件组成，并且大小远小于10 MB。

Sparse Indexing采用的是前缀特征采样算法，即选择那些前n位都为0的ChunkID作为Hook，当ChunkID随机均匀分布时，采样率为$1/2^n$。但由于ChunkID并不是完全均匀分布的，因此前缀特征采样是一种概率性的采样算法，不能保证采集到预期数量的Hook。如当Chunk的平均大小为μ KB，采样率取$1/2^n$时，从理论上讲，当Segment的大小达到$2^n \mu m$ KB(称为采样阈值)，就能够采样到m个Hook，但由于前缀特征采样算法是概率性的，因此并不能保证Hook数量达到预期值。一般而言Segment的大小超出阈值越多，采集到预期Hook数量的概率也越大。由于非传统备份负载的分段结果中，Segment普遍较小，因此很多Segment甚至1个Hook都采样不

到，这就使得这些Segment无法参与重复数据消除，导致很多重复Chunk无法被检测出，降低了重复数据消除性能。

2 多负载重复数据消除方法

针对上述前缀特征采样算法的缺点，本文提出一种最小特征采样算法，并在此基础上设计了一种适用于多种负载的重复数据消除方法MDM。

2.1 最小特征采样算法

设$\mathrm{Sgt_N}$的Chunk集合为$C_N=\{\mathrm{ck}_1,\mathrm{ck}_2,\cdots,\mathrm{ck}_k\}$，$|C_N|=k$，采样率为$R$，$h$为计算ChunkID的哈希函数，并且$h$近似满足min-wise independence条件[11]，$H(C_N)=\{h(\mathrm{ck}_1),h(\mathrm{ck}_2),\cdots,h(\mathrm{ck}_k)\}$，最小特征采样算法的步骤如下：

1) 计算特征采样数$m=\lfloor k\cdot R\rfloor$，如果$m\geqslant 1$，转到步骤3)，否则转到步骤2) 。

2) 从$H(C_N)$中取出最小的那一个ChunkID作为Hook，记为$H_N=\mathrm{Min}_1(H(C_N))$，转到步骤4) 。

3) 在$H(C_N)$中从小到大取出m个ChunkID作为Hook，记为$H_N=\mathrm{Min}_m(H(C_N))$，转到步骤4) 。

4) 采样算法结束。

最小特征采样算法是确定性的采样方法：当$m\geqslant 1$时，总能采样到恰好m个Hook；当$m<1$时，算法也保证了能够采样到1个Hook。换而言之，每个Segment都能采样到至少1个Hook，故所有的Segment都能参与重复数据消除。

如果两个Segment的相似程度越高，有效的采样算法应保证它们特征集的交集不为空的概率越高。下面对最小特征采样算法的有效性进行简要证明。

首先定义Segment的相似度：设C_1和C_2分别代表$\mathrm{Sgt_1}$和$\mathrm{Sgt_2}$的Chunk集合，则$\mathrm{Sgt_1}$和$\mathrm{Sgt_2}$的Jaccard相似度$\mathrm{Sim}(\mathrm{Sgt_1},\mathrm{Sgt_2})$[12]为：

定义 1 $$\mathrm{Sim}(\mathrm{Sgt_1},\mathrm{Sgt_2})=\frac{|H(C_1)\cap H(C_2)|}{|H(C_1)\cup H(C_2)|} \quad (1)$$

Broder扩展定理[11]保证了最小特征采样算法的有效性，该定理如下：

定理 1 两集合C_1和C_2，令$I=|C_1\cap C_2|$，$U=|C_1\cup C_2|$，$H_i=\mathrm{Min}_m(H(C_i))=\mathrm{Min}_m(\{h(x_j)\mid \forall x_j\in C_i\})$，$(i=1,2)$，$h$是满足min-wise independence条件的哈希函数，$\mathrm{Min}_m(H(C_i))$代表从集合$H(C_i)$中从小到大取出m个元素组成的集合，则有：

$$\Pr(H_1\cap H_2=\varnothing)\leqslant \frac{(U-I)(U-I-1)\cdots(U-I-m-1)}{U(U-1)\cdots(U-m-1)} \quad (2)$$

由于$\mathrm{Sim}(\mathrm{Sgt}_1,\mathrm{Sgt}_2)=I/U$，且$m$值相对于$U$值小得多，因此式(2)可以化简为：

$$\Pr(H_1\cap H_2=\varnothing)\leqslant(1-\mathrm{Sim}(\mathrm{Sgt_1},\mathrm{Sgt_2}))^m+\varepsilon \quad (3)$$

式中，ε是一个随着U值增大而迅速减小的误差因子。由式(3)可以得出如下推论：

推论 1 如果$\mathrm{Sgt_1}$和$\mathrm{Sgt_2}$的Jaccard相似度为s，用上述最小特征采样算法进行采样，则这两个分段的Hook集合交集不为ϕ的概率大于$1-(1-s)^m$。

由此可见，两个分段Hook集合交集非空的概率随着这两个分段相似度的增大而增大，且m值越大，增大的速度越快。因此，最小特征采样算法得到的特征集可以准确反映分段之间的相似度。

2.2 MDM的主要过程

在上述最小特征采样算法的基础上，设计了多负载重复数据消除方法MDM，该方法的主要流程如图1所示。下面重点说明该方法与Sparse Indexing方法不同的部分，对相似的部分不再赘述。

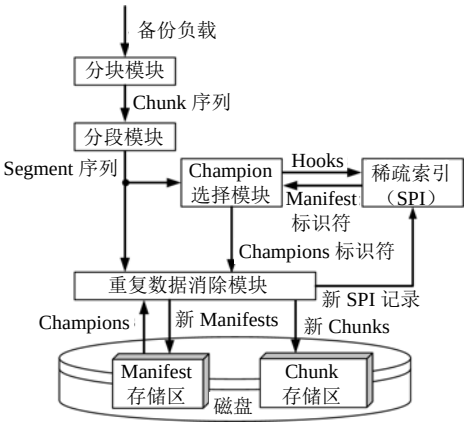

图1 MDM流程图

1) 对备份负载进行分块，分块时Chunk边界遵从于文件边界，即所有的Chunk均不跨越文件。

2) 对步骤1)中得到的Chunk序列进行分段。图2为Sparse Indexing和MDM两种方法分别对两种负载(基于数据流的传统备份负载和基于小文件的非传统备份负载)的预处理结果比较图，MDM对非传统负载的预处理结果与Sparse Indexing是相同的，故图中略去。可以看出，MDM的预处理结果中，所有的Chunk均不跨越文件边界，由于文件的数据变化常常发生在文件头部和尾部，因此可以减小数据变化所影响的范围，提高重复数据消除性能。

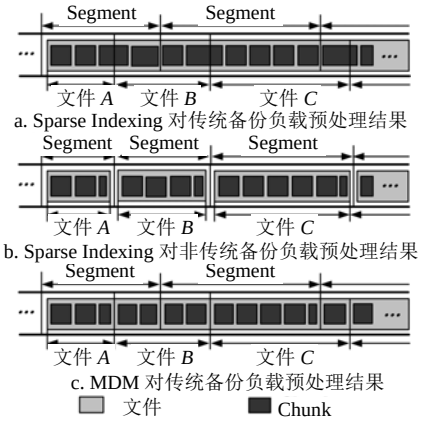

图2　负载预处理结果比较图

3) 对每个新到的$\mathrm{Sgt_N}$，用上述最小特征采样算法对其进行特征采样，得到其特征集$H_N=\{\mathrm{hk}_1,\mathrm{hk}_2,\cdots,\mathrm{hk}_m\}$。

4) 用H_N查询内存中的SPI，得到待选的ManifestID集合$M_N=\{\mathrm{mid}_1,\mathrm{mid}_2,\cdots,\mathrm{mid}_p\}$。SPI的结构如图3所示，SPI中仅保存Manifest的标识符。为控制SPI的大小，ManifestID链表的长度设置了一个上限值lmt，当某一Hook的ManifestID链表长度达到了lmt时，从中删除掉最旧的mid后再添加新的。生成M_N的过程形式化表达为：

$$M_N=\bigcup_{1\leqslant j\leqslant|H_N|}(\mathrm{SPI}(\mathrm{hk}_j)) \tag{4}$$

式中，$\mathrm{SPI}(\mathrm{hk}_j)$代表用$\mathrm{hk}_j$查询SPI得到的mid集合，例如查询如图3所示的SPI，则有$\mathrm{SPI}(d)=\{\mathrm{mid}_1,\mathrm{mid}_2,\mathrm{mid}_3\}$。$M_N$中每个元素$\mathrm{mid}_j$指向一个链表$\mathrm{Lst}_j$，该链表保存了$\mathrm{mid}_j$对应的分段清单$\mathrm{mf}_j$与$H_N$共享的Hook，即$\mathrm{Lst}_j=\mathrm{mf}_j\cap H_N$。

Hook	ManifestID
a	mid_1
b	mid_1 → mid_2
c	mid_1 → mid_2 → mid_3
d	mid_1 → mid_2 → mid_3
e	mid_3
f	mid_4
g	mid_4
	.....
m	mid_1 → mid_2
	.....

图3　SPI结构示意图

5) 为减少磁盘读写量，仅从M_N中选出t个mid组成ChampionID集合$P_N=\{\mathrm{pid}_1,\mathrm{pid}_2,\cdots,\mathrm{pid}_t\}$，选取的具体步骤如下：

① $P_N\leftarrow\phi$ //清空P_N集合。

② $\mathrm{pid}^*=\mathrm{ChooseCp}(\mathrm{Max}(\{|\mathrm{Lst}_i||1\leqslant i\leqslant|M_N|\}))$ //从M_N中选出与H_N共享Hook数量最多的那个mid(如果有多个mid并列第一，则选择存储时间最新的那一个)，记为pid^*。

③ $P_N\leftarrow P_N\cup\{\mathrm{pid}^*\}$，$M_N\leftarrow M_N-\{\mathrm{pid}^*\}$ //将pid^*添加到P_N中，并从M_N中删除pid^*及其对应的Lst。

④ if $|P_N|=t\,\|\,M_N=\phi$，goto 步骤8)// 如果ChampionID已经达到预期数量或者已经没有可选的ManifestID，就转到步骤⑧。

⑤ $\mathrm{Lst}_i\leftarrow\mathrm{Lst}_i-\mathrm{Lst}(\mathrm{pid}^*),1\leqslant i\leqslant|M_N|$ //将那些pid^*与H_N共享的Hook从M_N剩下的Lst中删除。

⑥ $M_N\leftarrow M_N-\{\mathrm{mid}_i\,|\,\mathrm{Lst}_i=\mathrm{null}\},1\leqslant i\leqslant|M_N|$ //将那些Lst为空的mid从M_N中删除。

⑦ 转到步骤②继续选取ChampionID。

⑧ 选取过程结束。

步骤⑤的目的在于使得I_h尽可能地大，从而提高重复数据消除性能。

$$I_h=\left|\left(\bigcup_{1\leqslant i\leqslant|P_N|}\mathrm{Hook}(\mathrm{pid}_i)\right)\cap H_N\right| \tag{5}$$

式中，$\mathrm{Hook}(\mathrm{pid}_i)$代表$\mathrm{pid}_i$对应的Hook集合。

6) 将P_N中各*pid*对应的Champion从磁盘调入内存作为检索源，对$\mathrm{Sgt_N}$进行重复数据消除。

7) 为$\mathrm{Sgt_N}$创建对应的分段清单$\mathrm{mf_N}$，并存储在磁盘上。

8) 如果$\mathrm{Sgt_N}$的特征采样数$m\geqslant1$，则用$\mathrm{Sgt_N}$的Hook更新SPI；反之则不更新SPI。有条件地更新SPI，可以把SPI有限的链表长度留给相对较大的Segment，因为它们包含更多的Chunk，这可以使得算法尽量选取较大的Segment作为检索源，有利于提高重复数据消除效果。

2.3　一个简化的实例

下面用一个简化的实例说明MDM中的关键步骤。设某$\mathrm{Sgt_N}$经过最小特征采样后得到的Hook集合H_N如图4a所示，经查询如图3所示的SPI后，得到待选ManifestID集合$M_N=\{\mathrm{mid}_1,\mathrm{mid}_2,\mathrm{mid}_3,\mathrm{mid}_4\}$，如图4b所示。这4个Manifest中与$H_N$相同的Hook在图4a中用阴影标识。选取ChampionID过程中M_N集合的变化情况如图4c所示，最终得到$P_N=\{\mathrm{mid}_1,\mathrm{mid}_4,\mathrm{mid}_3\}$，$I_h$=7；但如果选取ChampionID的过程中没有步骤⑤，则最后得到的$P_N'=\{\mathrm{mid}_1,\mathrm{mid}_2,\mathrm{mid}_3\}$，$I_h'$=5。从图中可以直观地看出，$P_N$包含了$\mathrm{Sgt_N}$的全部Hook，但$P_N'$却只包含

了Sgt_N的部分Hook，因此 P_N 对应的Manifest也比 P_N' 包含更多与Sgt_N相同的Chunk，即以 P_N 作为检索源可以检测出更多Sgt_N中的重复数据。

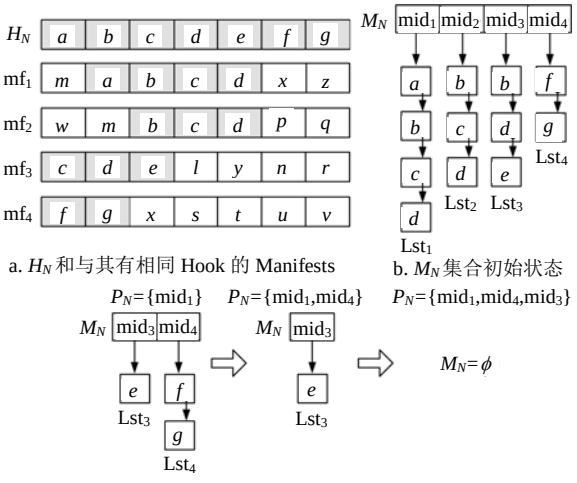

图4 MDM简化实例示意图

3 实验及分析

3.1 实验负载

为考察MDM处理混合备份负载的性能，本文采集了20个不同用户连续60天的备份文件作为实验数据集。采集的文件总数约为8.79×10^6个，总大小约为2 TB(2071.30 GB)，文件的平均大小约为247 KB。这些文件按以下方式组成实验负载：

1) 用Tar工具将第 j 天($1\leqslant j\leqslant 60$)的备份文件按不同用户分别链接成数据包，各数据包内文件按照其相对路径和文件名排序，再将这些数据包按用户编号 i ($1\leqslant i\leqslant 20$)排序，组成第 j 天的基于数据流的备份负载，如图5a所示。

2) 将所有用户第 j 天的备份文件按照备份的时间排序，形成第 j 天的基于小文件的备份负载，如图5b所示。

3) 将全部数据流负载和小文件负载按天数顺次排列，形成混合备份负载，如图5c所示。

混合备份负载既包含了传统的数据流负载，又包含了非传统的小文件负载，并且这两种负载之间有大量的重复数据。在实际应用中，这种情况是普遍存在的，例如同一个文件，可能既被平时的增量备份包含，又被每周的全备份包含，平时的增量备份采用小文件负载形式而全备份采用数据流负载形式。如果备份系统具有较强的跨负载重复数据检测能力，就可以节省更多的存储空间。

混合备份负载可以综合考察重复数据消除方法对多种负载的重复数据消除能力。

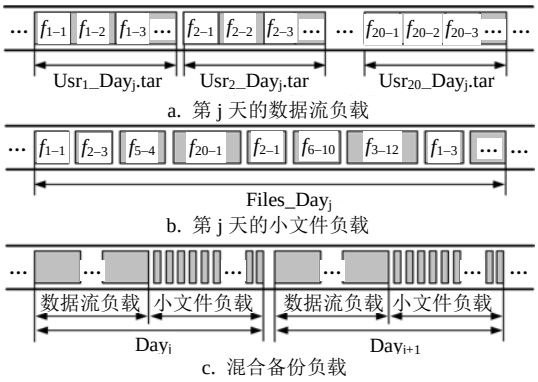

图5 实验备份负载示意图

3.2 实验结果

用压缩比 CR 来衡量重复数据消除性能，如式(6)所示，其中 $|\mathrm{DO}|$ 代表数据对象的原始大小，$|\mathrm{DO}|_d$ 代表数据对象经过重复数据消除之后的大小。

$$\mathrm{CR}=|\mathrm{DO}|/|\mathrm{DO}|_d \tag{6}$$

图6比较了如下几种方法的重复数据消除性能：

1) Flat Index: 用基于平面型索引的传统方法对混合备份负载进行重复数据消除。

2) Sparse Indexing-1: 用Sparse Indexing分别对数据流备份负载和小文件备份负载进行重复数据消除(处理两种负载时所使用的索引是相互独立的)，最终得到的总体压缩比。

3) Sparse Indexing-2: 用Sparse Indexing对混合备份负载进行重复数据消除。

4) MDM-1: 用改进的特征采样方法，但预处理方法未改进的MDM分别对数据流备份负载和小文件备份负载进行重复数据消除，得到的总体压缩比。

5) MDM-2: 用改进的特征采样方法，但预处理方法未改进的MDM对混合备份负载进行重复数据消除。

6) MDM-3: 用改进的预处理方法和特征采样方法的MDM对混合备份负载进行重复数据消除。

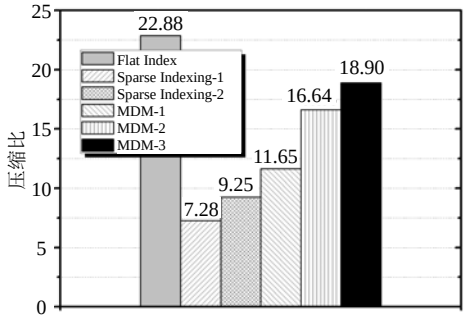

图6 重复数据消除性能比较

从图6中可以看出：

1) 由于基于Flat Index的方法可以检测出全部的重复Chunk，因此在同等情况下具有最优的重复数据消除性能。

2) 由于MDM可以有效处理小文件备份负载，因此它对混合备份负载的重复数据消除性能显著优于Sparse Indexing，MDM-3的压缩比相对于Sparse Indexing-2提高了1.04倍。

3) 由于可以跨负载检测重复数据，因此Sparse Indexing-2/MDM-2相对于Sparse Indexing-1/MDM-1压缩比均有提升(分别提升了27.06%和42.83%)，并且MDM-2相对于MDM-1的提升更加显著，这说明MDM具有更强的跨负载重复数据消除能力。

4) MDM-3的压缩比相对于MDM-2进一步提高了13.58%，说明Chunk不跨越文件边界有利于提高重复数据消除性能。

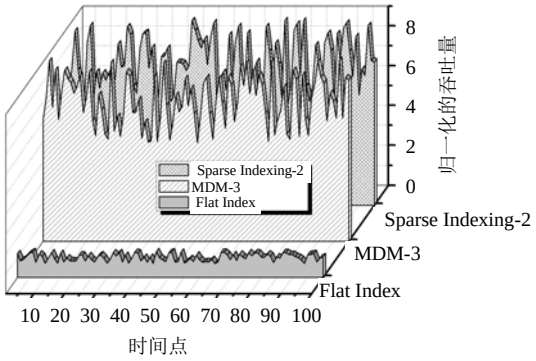

图7　吞吐量比较

图7为在100个时间点上比较几种方法的瞬时吞吐量(算法单位时间内处理的负载量)，为了便于比较，图中的吞吐量值均以Flat Index的平均吞吐量为单位进行了归一化处理，可以看出：

1) 在数据规模较大时，基于Flat Index的方法由于需要频繁在内存和磁盘之间交换索引分页，因此吞吐量性能很差。

2) Sparse Indexing和MDM由于都只在内存中维护了完整索引的一个很小的子集，并通过批量调入ChunkID，摊销了磁盘I/O的开销。因此它们的吞吐量远高于基于Flat Index的方法，二者的平均吞吐量分别为Flat Index的7.14倍和7.01倍。

3) MDM由于改进了特征采样方法，因此需要处理更多的Hook，故吞吐量略低于Sparse Indexing，其平均值为Sparse Indexing的98.18%。

4　结　论

Sparse Indexing方法利用数据的局部性，可有效提高处理传统备份负载时的吞吐量并保持较优的重复数据消除性能，但不能有效处理基于小文件的非传统备份负载。针对该问题，本文提出了一种多负载重复数据消除方法MDM，该方法改进了Sparse Indexing的特征采样算法和负载预处理方法，使其在保留原有优点的同时，可以有效处理小文件备份负载。实验结果表明，该方法可以很好地处理混合备份负载和消除跨负载的重复数据，从而在吞吐量相对于Sparse Indexing近似不变的情况下，压缩比提高到Sparse Indexing的2.04倍。该方法适用于需要处理多种类型备份负载的高性能重复数据消除系统。

参　考　文　献

[1] GANTZ J F, CHUTE C, MANFREDIZ A, et al. The diverse and exploding digital universe: an updated forecast of worldwide information growth through 2011[R]. Framingham, USA: International Data Corporation, 2008.

[2] MEYER D T, BOLOSKY W J. A study of practical deduplication[C]//The 9th USENIX Conference on File and Storage Technologies. Berkeley, USA: USENIX Association, 2011.

[3] CONSTANTINESCU C, GLIDER J, CHAMBLISS D. Mixing deduplication and compression on active data sets[C]//The 2011 Data Compression Conference. New York,USA: IEEE Computer Society, 2011.

[4] 敖莉, 舒继武, 李明强. 重复数据删除技术[J]. 软件学报, 2010, 21(5): 916-929.
AO Li, SHU Ji-wu, LI Ming-qiang. Data deduplication techniques[J]. Journal of Software, 2010, 21(5): 916-929.

[5] 王灿, 秦志光, 冯朝胜, 等. 面向重复数据消除的备份数据加密方法[J]. 计算机应用, 2010, 30(7): 1763-1766.
WANG Can, QIN Zhi-guang, FENG Chao-sheng, et al. Deduplication-oriented backup-data encryption method[J]. Journal of Computer Applications, 2010, 30(7): 1763-1766.

[6] LILLIBRIDGE M, ESHGHI K, BHAGWAT D, et al. Sparse indexing: large scale, inline deduplication using sampling and locality[C]//The 7th USENIX Conference on File and Storage Technologies. Berkeley, USA: USENIX Association, 2009:111-123.

[7] KRUUS E, UNGUREANU C, DUBNICKI C. Bimodal content defined chunking for backup streams[C]//The 8th USENIX Conference on File and Storage Technologies. Berkeley,USA: USENIX Association, 2010:18-31.

[8] ESHGHI K, TANG H K. A framework for analyzing and improving content-based chunking algorithms[R]. Palo Alto: Hewlett-Packard Labs, 2005.

[9] CLEMENTS A T, AHMAD I, VILAYANNUR M, et al. Decentralized deduplication in SAN cluster file systems[C]//The USENIX Annual Technical Conference. Berkeley, USA: USENIX Association, 2009:101-114.

[10] BINGHAM S F, BUCHMAN M D, SINGHAL U, et al. Network configuration backup and restore operations using continuous data protection: US Patent, 7971091[P]. 2011-06-28.

[11] BRODER A Z, CHARIKAR M, FRIEZE A M, et al. Min-wise independent permutations[J]. Journal of Computer and System Sciences, 2000, 60(3): 630-659.

[12] BRODER A Z. On the resemblance and containment of documents[C]//The Compression and Complexity of Sequences 1997. New York, USA: IEEE Computer Society, 1997: 21-29.

编　辑　税　红