基于存储环境感知的重复数据删除算法优化

周敬利 聂雪军 秦磊华 刘 科 朱建峰 王 宇
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 在备份和归档等存储应用中产生的重复数据在存储空间和能耗上造成的浪费问题日益突出,如何删除重复数据已成为当前存储领域中的研究热点。CDC(Content-Defined Chunking)是一种适用于多种应用环境的重复数据删除算法,但缺乏针对具体应用环境的优化。通过对存储环境进行感知和分析,为CDC算法提出了两个参数选择约束条件:(1)根据存储设备中的数据块存储方式来选择平均分块大小等参数;(2)根据数据的分块边界分布特性来选择分块边界特征值参数。实验表明,与无约束条件的CDC算法相比,这两个约束条件在4个实验数据集上平均可提高16.3%的数据缩减比。

关键词 重复数据删除,存储环境感知,CDC,文件系统,分块边界
中图法分类号 TP334.5 **文献标识码** A

Optimization for Data De-duplication Algorithm Based on Storage Environment Aware

ZHOU Jing-li NIE Xue-jun QIN Lei-hua LIU Ke ZHU Jian-feng WANG Yu
(College of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074, China)

Abstract Storage applications such as backup and archive are creating more and more duplication data, which has caused increasing waste in storage space and energy consumption, and how to delete duplication data has become a hot subject in research. CDC(Content-Defined Chunking) is a prevail algorithm for data de-duplication and can be applicable in various environment, however it does not take into account some characteristics which are specific to individual environment and can influence its result. We studied the CDC's application in storage system and put up with two constraints for determining parameters for CDC: (1)Determining parameters such as average block size based on the block organization of storage devices; (2)Determining block boundary based on candidate boundary distribution. The result indicates that, comparing with the separate CDC without restraint conditions, these two constraints can achieve 16.3% higher compression ratio with 4 data sets.

Keywords Data de-duplication, Storage environment aware, CDC, File system, Block boundary

1 引言

在备份和归档等存储应用中产生的重复数据可以占到总存储容量的90%以上[1]。重复数据不仅消耗了不必要的磁盘空间和网络带宽,而且在能耗和冷却等方面带来的问题也日益突出。因此,如何删除重复数据已成为存储领域中的研究热点。

CDC是一种可适用于多种应用环境的重复数据删除算法,包括存储系统[1-9]、网络文件系统[10,11]等。在这些系统中,首先通过CDC算法将文件分割为变长的数据块,然后通过MD5或者SHA-1等算法为每个数据块计算出一个惟一的数字指纹,并根据数字指纹来判断系统是否已经存在重复的数据块,只有当数据块不重复时才会被保存到存储设备上。在CDC算法中需要预先选择一组参数:分块边界特征值、平均分块大小、分块大小的最小值和最大值以及滑动窗口的大小。然而,由于CDC算法的广泛适用性,它在应用于具体环境时没有考虑不同环境的特性对算法效率的影响,因此在针对具体应用环境进行优化上存在着不足。

针对以上问题,本文以CDC在存储系统中的应用为背景,通过分析存储系统对CDC算法的影响因素,为算法提出了两个约束条件:(1)存储设备上数据的组织方式,即通过向存储设备发送查询命令来感知数据块存储方式以及逻辑块大小等参数,并以此来选择CDC算法中的平均分块大小、分块大小的最大值和最小值以及滑动窗口大小;(2)数据的分块边界分布特性,即通过分析不同类型存储数据对算法的影响来选择最优的分块边界特征值。我们在实验中选择了4个数据

到稿日期:2010-03-05 返修日期:2010-06-03 本文受部委基金"基于服务定制的智能存储系统研究",国家自然科学基金项目(606730001),国家"973"重点基础研究发展规划基金项目(2004CB318203)资助。

周敬利(1946—),教授,博士生导师,主要研究方向为高性能存储技术及系统、多媒体数据处理与通讯、网络安全;聂雪军(1979—),博士生,主要研究方向为内容感知存储系统和网络存储,E-mail:niexuejun@sina.com(通信作者);秦磊华(1968—),博士,副教授,主要研究方向为网络安全和网络存储;刘 科(1979—),博士生,主要研究方向为内容感知存储系统、存储数据检索;朱建峰(1978—),博士生,主要研究方向为网络存储系统、文件系统、智能容灾存储系统;王 宇(1975—),博士生,主要研究方向为网络存储系统、存储协议。

集，分别模拟源代码备份、电子邮件备份、网页备份以及音频文件备份等4种常见的存储应用。实验结果表明，与无约束条件的CDC算法相比，这两个约束条件在4个数据集上平均可使CDC提高16.3%的数据缩减比。

本文第2节介绍了CDC算法的基本流程及存在的问题；第3节和第4节分别给出了存储设备上的数据组织形式和数据的分块边界分布特性这两个约束条件；第5节介绍了实验环境与结果分析；最后总结了本文的工作以及下一步的研究方向。

2 CDC算法分析

CDC算法的基本流程为：首先选择一个长度为 w 的滑动窗口，然后将窗口在文件的数据流中逐字节推进；每当推进一个字节，将窗口中的当前字节计算出一个候选边界 f；然后判断候选边界是否符合预先设定的边界特征值，如果满足则被作为一个分块边界，两个分块边界之间的所有字节即构成一个数据块。

CDC中的候选边界计算方法[12,13]为：假定滑动窗口可以表示为一个长度为 w 的字节序列 $(t_1, t_2, \cdots, t_w)$，那么第1个候选边界的计算公式为：

$$f_1 = \sum_{i=0}^{w-1} (t_i \times 2^{w-1-i}) \bmod 2^w$$

在计算出 f_1 之后，第2个候选边界的计算公式为：

$$f_2 = (p * f_1 + t_{w+1} - t_0 * 2^{w-1}) \bmod 2^w$$

依次类推，后一个候选边界可以基于前一个候选边界的值来计算，因此算法的复杂度为 $O(N)$。

每计算一个候选边界 f，CDC都将判断确定 f 是否是一个分块边界：

如果 f 的后 k 位等于某个特征值 a，其中 a 的取值范围为 $[0, 2^k-1]$，那么 f 就是一个分块边界，所有基于 a 计算得到的分块边界构成一个集合 $A_a = \{f_{a1}, f_{a2}, \cdots, f_{am}\}$。

CDC在选择特征值 a 时假设：基于范围 $[0, 2^k-1]$ 内的任何一个取值计算得到的分块边界在数量上是基本相等的，即 a 在 $[0, 2^k-1]$ 上服从独立且均匀的概率分布，因此CDC对所有类型的数据都采用单一特征值 a，通常为0。

从上述分析可以看出，在CDC算法中需要指定的参数包括 w，a 和 k，其中 k 可以通过平均分块大小 S_{ave} 计算得到，即 $k = \log_2 S_{ave}$。此外，为了防止产生过大或者过小的数据块，CDC还需要指定分块大小的最小值 S_{min} 和最大值 S_{max}。

当CDC在应用于存储系统时，可以从两个方面来加以约束：

(1)使CDC分割得到的变长数据块与底层数据块的存储方式相匹配。在底层存储设备的文件系统中，通常以固定大小的逻辑块为存储单位，因此在将变长数据块存储到底层设备上时需要考虑一些因素：如果 S_{ave} 对逻辑块大小的取余运算结果越小，那么在逻辑块中平均所需的填充字节也就越多，相应地存储空间浪费也就越多；如果 S_{ave}，S_{min}，S_{max} 和 w 越大，那么生成的数据块也就越大，而填充字节所占的比例将降低，但文献[6,9]指出，数据块越大，CDC算法在识别重复数据块上的效率越低。

(2)为不同类型的存储数据选择不同的边界特征参数 a。CDC假设 a 在 $[0, 2^k-1]$ 上服从独立且均匀的概率分布，根据候选边界的计算公式，可由这个假设进一步推出：数据中的单个字节需在 $[0, 255]$ 上服从独立且均匀的概率分布，但这个条件并非在所有类型的数据中都成立。例如，在ASCII编码的文本数据流中，字节是在 $[0, 127]$ 上服从的独立且均匀的概率分布，而在Unicode-16编码的数据流中，每两个字节之间存在着关联而并非独立的。因此，对于不同类型的存储数据采用单一的 a 不能获得最优的效果。

基于上述分析，我们从存储设备上的数据块的存储方式和数据的分块边界分布特性出发，分别提出了2个约束条件来解决上述问题。

3 第1个约束条件：存储设备上的数据组织形式

DDFS(De-Duplication File System)是位于存储设备上的一个微文件系统，它是在文件系统ext2的基础上通过扩展和修改而形成的，负责数据块的存储和管理工作。在DDFS中，每个文件被表示为一组Hash值，每个Hash值对应于一个不重复的数据块；多个变长的数据块被保存在定长的“容器”，中以减少由于填充字节而造成的空间浪费。DDFS扩展了ext2中的超块结构，增加了Hash表信息和容器信息两项内容。在索引节点表后面增加了两块存储区，分别存放数据块Hash表和容器表。之后是目录数据区和存储数据块的容器区。DDFS的整体结构如图1所示。

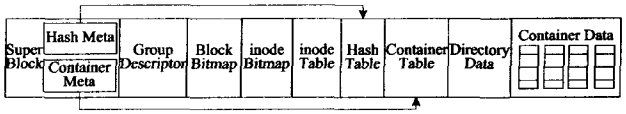


图1 DDFS的整体结构

3.1 相关数据结构

在DDFS中增加和修改的数据结构包括Hash表、容器以及索引节点等。

Hash表

Hash表由Hash表元数据与Hash表项构成。在Hash表元数据中包含以下域：Hash表起始偏移、Hash表中表项的总量、空闲表项的数量、空闲表项链表起始偏移、下一个空闲表项的索引等。每个Hash表项对应于一个数据块，Hash表项的结构如图2所示。其中，引用计数 *refcount* 表示数据块的重复数量，在删除文件时，与文件对应的所有数据块的引用计数都将减1，如果引用计数为0，则数据块将从存储设备中删除。

容器

容器由容器元数据和容器数据构成。在容器元数据中包含以下域：容器大小、容器中的数据块最大数量、容器总数量、剩余空间表起始偏移、容器数据区起始偏移。容器是一个固定大小的逻辑块，每个容器由容器头和数据区组成。在容器头中包含一组偏移 *offset*，表示每个数据块在容器中的起始位置。容器的结构如图2所示。

索引节点

在DDFS中，文件索引节点中逻辑块号被替换为数据块的Hash表项索引，其他域则保持不变。

索引节点、Hash表、容器的结构及数据块寻址方式如图2所示。数据块的寻址方式为：(1)根据文件名找到对应的索引节点；(2)读取数据块的Hash表项信息；(3)从表项中读取

数据块所在容器编号并计算容器的起始地址；(4)从 Hash 表项中读取数据块在容器中的偏移；(5)将容器起始地址和偏移值相加得到数据块的起始地址。

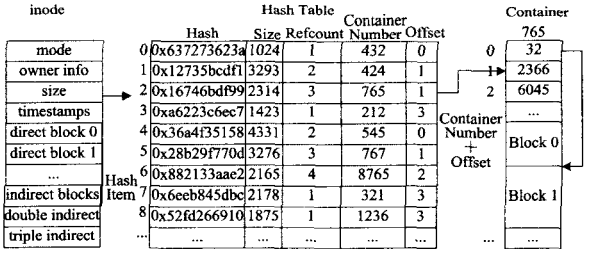


图 2　索引节点，Hash 表，容器的结构

3.2　存储空间分配算法

在 DDFS 中，每个数据块都是存储在容器中的。在容器剩余空间表中记录了每个容器中的剩余空间 *remain*。当 DDFS 初始化时，剩余空间表中的所有项将构成两个分配器：前端分配器 *front_allocator* 和后端分配器 *back_allocator*。在 *front_allocator* 中包含了所有已经存储了部分数据块且 $remain > S_{min} + sizeof(offset)$ 的容器，并根据每个容器的 *remain* 值在内存中构成一个递增的双向链表。在 *back_allocator* 中包含的是所有的尚未存储数据块的容器，并根据容器号的顺序构成一个递增的单向链表。存储空间分配的算法如下。

算法 1　存储空间分配

输入：数据块 *block*

输出：容器号 *cn* 及偏移号 *on*

1. 计算数据块在容器中所需的总字节数：
   $S_{block} = sizeof(block) + sizeof(offset)$
2. 如果 S_{block} 大于 *front_allocator* 中最后一个节点的 *remain*，则表示当前的所有容器都无法容纳这个数据块，因此从 *back_allocator* 中取下第一个节点 *node*，然后跳转到步骤 4；
3. 通过二分搜索法找到满足 $S_{block} < node.remain$ 且 *remain* 为最小的节点；
4. 从节点中取出容器号 *cn*，然后在 *cn* 的容器头中分配一个偏移号 *on*，如果 *on* 是第一个偏移，那么 *on* 位置上的偏移值为容器头的大小 $sizeof(container\_head)$，否则 *on* 位置上的偏移值为 $offset_{on} = offset_{on-1} + S_{block}$；
5. 调整节点的剩余空间大小 $node.remian = node.remain - S_{block}$，并重新对 *front_allocator* 排序；
6. 返回容器号 *cn* 及偏移号 *on*；

3.3　参数选择

假定 DDFS 中容器的大小为 S_C，容器能够容纳的数据块最大数量为 N_C，容器中每个偏移的大小为 S_{offset}，那么根据 DDFS 可以选择以下参数：

平均分块大小 S_{ave}

$$S_{ave} = (S_C - N_C \times S_{offset}) / N_C$$

分块大小最大值 S_{max}

最大分块大小不应该超过 S_C，极端情况就是在容器中只容纳一个数据块：

$$S_{max} = S_C - N_C \times S_{offset}$$

分块大小最小值 S_{min}

每个数据块需要一组元数据来表示，因此 S_{min} 不能超过所有元数据的总和，否则反而会浪费存储空间。在 DDFS 中，每个数据块需要的元数据为：

$$S_{meta} = S_{inode} + S_{hash} + S_{offset}$$

其中，S_{inode} 为数据块在索引节点中的信息所占字节，S_{hash} 表示 Hash 表项的大小。此外，S_{min} 为 2 的幂将有利于数据在内存中的对齐和计算。因此，S_{min} 的选取策略为：

$(S_{min} > S_{meta}) \&\& (reside(\log_2 S_{min}) == 0)$ 滑动窗口大小 w

两个滑动窗口即为一个最小分块，因此

$$w = S_{min}/2$$

在 DDFS 中，N_C 和 S_{offset} 被设定为固定值，这样容器的大小 S_C 仅依赖于平均分块大小 S_{ave}。从其他参数的选择条件同样可以推断出 S_{max}，S_{min} 与 w 等参数同样仅依赖于 S_{ave}。因此，约束条件 DDFS 中惟一的可变参数即为 S_{ave}。

4　第 2 个约束条件：数据的分块边界分布特性

CDC 在选择边界特征值 a 时假设基于范围 $[0, 2^k - 1]$ 内的任何一个取值计算得到的分块边界在数量上是基本相等的。然而，通过对不同存储应用的数据进行计算，我们发现这个假设并非在所有情况下都成立。为了表示不同特征值 a 计算得到的分块边界分布形态，本文提出了分块边界直方图的概念。

定义 1(分块边界直方图)　以边界特征值 a 为观测对象，统计 a 在观测范围 $[0, 2^k - 1]$ 内的每个取值所对应的分块边界数量，所形成的直方图即为分块边界直方图 $F = \{p_0, p_1, \cdots, p_{2^k-1}\}$，其中 P_i 表示当 $a = i$ 时的分块边界数量。

根据定义 1，每个文件 d 都可以计算出一个 F_d，而文件集合 D 中所有 F_d 的叠加就构成了文件集合的分块边界直方图 $F_D = \sum_{d \in D} F_d$。

在图 3(a)—(f)中分别给出了 html，email 和 mp3 这 3 种文件类型中每种类型的单个文件分块边界直方图与文件集合分块直方图。从图 3 中可以看出：(1)不同类型文件的分块边界直方图有着不同的形态，只有 mp3 类型的分块边界直方图最接近 CDC 算法的假设，而在 html 和 email 等类型中算法假设并不成立；(2)在同一类型的文件中，单个文件的 F_d 与文件集合的 F_D 非常类似，因此可以认为，基于 F_D 计算得到的边界特征值从平均意义上来看也是单个文件的最优边界值或者接近于最优边界值。

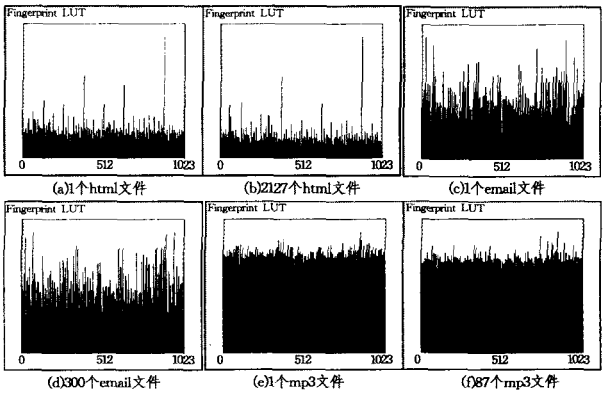


图 3　html，email 以及 mp3 等 3 种文件类型的单个文件分块边界直方图与文件集合分块边界直方图

基于上述分析，我们提出了根据第 2 个约束条件来选择 CDC 中边界特征值参数的算法

算法 2 边界特征值的选择

输入：样本文件集合 $D=\{d_0,d_1,\cdots,d_n\}$，平均分块大小为 S_{ave}，滑动窗口大小 w

输出：边界特征值 a

1. 计算出样本文件集中每个文件的分块边界直方图，$F_{d_i}=\{p_{i0},p_{i1},\cdots,p_{i(2^k-1)}\}$；
2. 通过 $\sum_{d\in D}F_d$ 计算样本集合的直方图 $F_D=\{p_0,p_1,\cdots,p_{2^k-1}\}$；
3. 计算分块的数量：$C=(\sum_{0}^{2^k-1}p_i+w-1)/S_{ave}$；
4. 设置一个分块数量变动率 r，并构造区间 $R=[C\times(1-r),C\times(1+r)]$；
5. 构造候选特征集合 $K=\{a\mid C\times(1-r)\leqslant p_a\leqslant C\times(1+r)\}$；
6. 以集合 K 中的每个元素为边界特征值对文件进行分块，计算实际的平均分块大小；
7. 以最接近 S_{ave} 的分块边界作为边界特征值 a。

5 实验结果与分析

5.1 评价指标

在多数文献[1,4-9,11]中都采用了数据缩减比 C_D 来评价重复数据删除算法的性能：

$$C_D=\frac{D_l}{D_p}$$

式中，D_l 和 D_p 分别表示在删除重复数据之前和之后的存储空间大小。

在实验中将计算 4 个缩减比，即 CDC 分别在无约束条件、仅第 1 个约束条件、仅第 2 个约束条件以及所有 2 个约束条件等情况下的缩减比。

5.2 实验数据

由于目前还无法获得真实的企业备份数据，因此我们收集了一些有代表性的数据集来模拟常见的备份/归档应用。在文献[1,6,7,9]中使用的实验数据类型包括源代码、数据库文件、二进制程序、网页、电子邮件以及音频文件。因此，我们在实验中使用了以下 4 个数据集：

数据集 1

名称：Linux Kernel 源代码

来源：http://www.kernel.org

说明：共下载了 2.6.12－2.6.18 共 7 个版本的源代码，包括文件 131,596 个，数据总量为 4902.0MB。

数据集 2

名称：TREC 2008 网页数据集

来源：http://es.csiro.au/cerc/data

说明：从科研机构 CSIRO 网站（http://www.csiro.au）上收集到的网页数据，共分为 267 个子集（000－266），409,551 个文件，数据总量为 4499.8MB。

数据集 3

名称：TREC 2005 邮件数据集

来源：http://plg.uwaterloo.ca/~gvcormac/treccorpus/

说明：从科研机构 CSIRO 内部收集的邮件数据，共分为 307 个子集（CSIRO000-CSIRO306），92,189 封邮件，数据总量为 3464.5MB。

数据集 4

名称：mp3 文件

来源：http://www.diggcd.com/

说明：从音乐下载网站收集的 mp3 文件，64 张专辑，包括 1,005 个文件，数据总量为 4614.5MB。

5.3 实验结果及分析

我们首先需要从每个数据集中分别选取一个样本数据集合并计算集合的边界特征值。样本数据集的选取方式如表 1 所列。

表 1 样本数据集

数据集	样本数据集
1	2.6.12 版本代码中 Driver 目录下的所有文件，共 4360 个文件，192.3MB
2	数据集 CSIRO000，共 2127 个文件，16.4MB
3	数据集 000，共 300 个文件，9.5MB
4	6 张专辑，共 87 个文件，490.1MB

在约束条件 1 中指出，DDFS 中惟一的可变参数为 S_{ave}，因此在实验中计算 4 个样本数据集在不同 S_{ave} 下的特征边界值，如表 2 所列。

表 2 样本数据集在不同 S_{ave} 下的特征边界值

S_{ave}	256	512	1024	2048	4096
1	0xB7	0x137	0x337	0xF7	0x5BF
2	0xC1	0xB7	0x2FE	0x7F5	0x544
3	0xFB	0x15A	0x32B	0x51B	0x242
4	0xB7	0x1C7	0x317	0x76F	0xF6F

在计算出边界特征值后，4 个数据集在不同约束条件下的缩减比如图 4－图 7 所示。

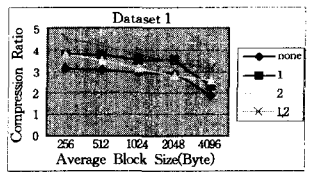


图 4 数据集 1 的缩减比

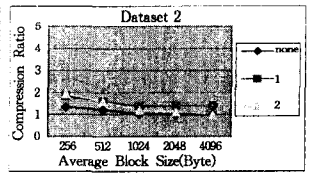


图 5 数据集 2 的缩减比

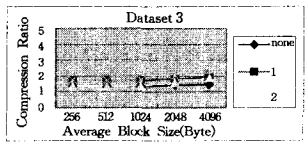


图 6 数据集 3 的缩减比

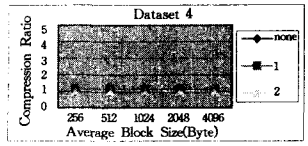


图 7 数据集 4 的缩减比

从实验结果中可以得出以下结论：

(1)约束条件 1 可以提高数据集的缩减比，但其效果随数据集的不同而不同。数据集 1,2,3 的效果较为明显，而数据集 4 的效果不明显。这是因为数据集 1,2,3 中的文件较小，因此填充空间所占比例较大，而约束条件 1 的作用正是可以减少填充空间。数据集 4 中的文件较大，因此可减少的填充空间所占比例较小。

(2)约束条件 2 在多数情况下可以提高数据集的缩减比，但在少数情况下也可能降低缩减比，例如在数据集 1，平均分块大小为 2048 字节的情况下，数据集本身的数据分块特性与字节构成有关。同样，约束条件 2 在数据集 1,2,3 上的效果比数据集 4 上效果要明显。

(3)在 4 个数据集上，约束条件 1 平均可以提高 12.8% 的缩减比，约束条件 2 平均可以提高 6.9%的缩减比，而 2 个约束条件共同可以提高 16.3%的缩减比。

结束语 本文通过对存储环境的感知和分析，从数据块组织方式以及数据分块边界分布特性等方面为重复数据删除算法 CDC 提出了两个参数选择约束条件。实验表明，在 4 个实验数据集上，约束条件 1 在所有情况下都能够提升 CDC 的缩减比，约束条件 2 则在大多数情况下可以提升 CDC 的缩减

比，如果同时应用两个约束条件则可以平均提升 16.3%的缩减比。

在接下来的工作中，我们将在以下方面做进一步研究：(1)扩大存储环境的感知范围，例如通过感知网络特性来分析网络存储环境下优化 CDC 算法的约束条件；(2)分析约束条件 2 中降低缩减比的原因，并对边界特征值选择算法进行调整。

参 考 文 献

[1] Zhu B, Kai L, Patterson H. Avoiding the disk bottleneck in the data domain deduplication file system[C] // 6th USENIX Conference on File and Storage Technologies(FAST'08). San Jose, CA, USA, 2008

[2] Cox L P, Murray C D, Noble B D. Pastiche: making backup cheap and easy [C] // Proceedings of the Fifth Symposium on Operating Systems Design and Implementation(OSDI'02). Boston, MA, USA, 2002

[3] Quinlan S, Dorward S. Venti: a new approach to archival storage [C]// Proceedings of the Conference on File and Storage Technologies(FAST'02). Monterey, CA, USA, 2002

[4] Douglis F, Iyengar A. Application-specific delta-encoding via resemblance detection [C] // General Track 2003 USENIX Annual Technical Conference. San Antonio, TX, USA, 2003

[5] Jain N, Dahlia M, Tewari R. TAPER: tiered approach for eliminating redundancy in replica synchronization [C] // 4th USENIX Conference on File and Storage Technologies(FAST'05). San Francisco, CA, USA, 2005

[6] Bobbarjung D R, Jagannathan P, Dubnicki P. Improving duplicate elimination in storage systems [J]. ACM Transactions on Storage(TOS), 2006, 2(4): 25

[7] Chuanyi L, Dapeng J, Yu G, et al. Semantic data de-duplication for archival storage systems [C]// 13th Asia-Pacific Computer Systems Architecture Conference(ACSAC'08). Hsinchu, China, 2008

[8] Lillibridge M, Eshghi K, Bhagwat D, et al. Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality [C] // FAST 09. 2009

[9] You L L, Karamanolis C. Evaluation of Efficient Archival Storage Techniques [C]// Proceedings of the 21st IEEE Symposium on Mass Storage Systems and Technologies(MSST'04). 2004

[10] Muthitacharoen A, Chen B, Mazieres D. A low-bandwidth network file system[J]. Operating Systems Review(ACM), 2002, 35(5): 174-187

[11] Hong B, Plantenberg D, Long D D E, et al. Duplicate Data Elimination in a SAN File System [C] // Proceedings of the 21st IEEE/12th NASA Goddard Conference on Mass Storage Systems and Technologies(MSST'04). 2004

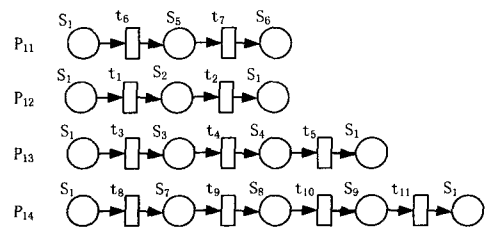
[12] Rabin. Fingerprinting by random polynomials [R]. 1981

[13] Manber U. Finding similar files in a large file system [C]// Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, CA, USA, 1994

[14] Moodalbail G, Mandagere N, Raghuveer A, et al. Backup aware object based storage[R]. 2007-25. DTC Intelligent Storage Consortium, June 2007

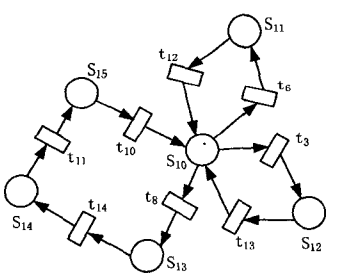
[15] Mesnier M, Ganger G R, Riedel E. Object-based storage [J]. IEEE Communications Magazine, 2003, 41(8): 84-90

(上接第 58 页)

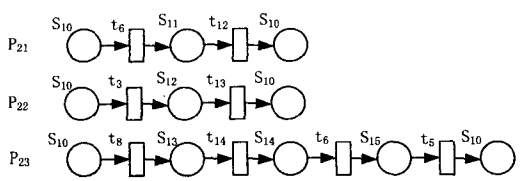


(b) Σ_1 的基本子进程集

图 5 有界 Petri 网 Σ_1 及其基本子进程集



(a) 有界 Petri 网 Σ_2



(b) Σ_2 的基本子进程集

图 6 有界 Petri 网 Σ_2 及其基本子进程集

结束语 文献[1]中提出了有界 Petri 网进程表达式的概念，即一个有界 Petri 网的进程表达式是以该网的基本子进程集为字母表的正规表达式。但在现有的文献中并没有给出有效求解基本子进程的方法。本文的贡献之一是提出了稳态网的概念并给出了一个算法，根据该算法可以求解有界稳态网的基本子进程集，然后根据所求得的基本子进程集来构造该网的进程表达式。本文的另外一个贡献是给出了由符合一定条件的 S-网的进程表达式来构造其同步合成网进程表达式的算法。进程表达式全面反映了网系统运行的各种可能，为 Petri 网分析提供了很好的途径。

参 考 文 献

[1] Wu Zhe-hui. Process expression of bounded Petri net[J]. Science in China(Series E), 1996, 39(1): 37-49

[2] 吴哲辉，王培良，赵茂先. 无界公平 Petri 网的进程表达式[J]. 计算机学报，2000，23(4)：337-344

[3] 曾庆田，吴哲辉. Petri 网的进程网系统[J]. 计算机学报，2002，25(12)：1308-1315

[4] 吴哲辉. Petri 网导论[M]. 北京：机械工业出版社，2006

[5] Hopcroft J E, Ullman J D. Introduction to Automata Theory, Languages, and Computation [M]. Addison-Wesley Publishing Company, 1979

[6] Latteux M, Roos Y. Synchronized shuffle and regular languages [M]. Jewels are Forever-Contributions on Theoretical Computer Science in Honor of Arto Salomaa, Berlin: Springer-Verlag, 1999: 35-44

[7] 蒋昌俊. 离散事件动态系统的 PN 机理论[M]. 北京：科学出版社，2000