DOI:10.16660/j.cnki.1674-098x.2008.21.021

哈夫曼编码应用的一种改进

——范式哈夫曼编码

邵天增[1] 尚冬娟[2]
(1.华东师范大学 上海 200062; 2.运城学院计算机科学与技术系 山西运城 044000)

摘 要:哈夫曼编码是一种变长编码,一种最优前缀编码技术,其实现了数据压缩,但其存在的不足直接制约了它的广泛应用。本文主要介绍一种改进方法——范式哈夫曼编码及译码算法,以解决其应用的不足。

关键词:哈夫曼编码 范式哈夫曼编码 范式哈夫曼译码
中图分类号:TP32 文献标识码:A 文章编号:1674-098X(2008)07(c)-0029-02

1 引言

哈夫曼编码是Huffman在上世纪五十年代初提出的,根据字符出现的概率来构造平均长度最短的编码,是一种变长编码。哈夫曼编码是哈夫曼树的一个应用,是一种最优的前缀编码技术,然而其存在的不足却制约了它的直接应用。首先,其解码时间为O(lavg),其中lavg为码字的平均长度;其次,更为最重要的是,解码器需要知道哈夫曼编码树的结构,因而编码器必须为解码器保存或传输哈夫曼编码树。对于小量数据的压缩而言,这是很大的开销。因而,应用哈夫曼编码的关键是如何降低哈夫曼编码树的存储空间。本文主要介绍一种常用的方法——范式哈夫曼编码以及解码算法,目前流行的很多压缩方法都使用了该技术,如GZIB、ZLIB、PNG、JPEG、MPEG等。

2 哈夫曼编码

2.1 哈夫曼树

哈夫曼树又称最优二叉树,是一种带权路径长度最短的二叉树。所谓树的带权路径长度,就是树中所有的叶结点的权值乘上其到根结点的路径长度。树的带权路径长度记为WPL=(W1×L1+W2×L2+W3×L3+...+Wn×Ln),N个权值Wi(i=1,2,...n)构成一棵有N个叶结点的二叉树,相应的叶结点的路径长度为Li(i=1,2,...n)。可以证明哈夫曼树的WPL是最小的。

怎样构造一棵哈夫曼树呢?最具有一般规律的构造方法就是哈夫曼算法:

1)对给定的n个权值{W1,W2,W3...,Wi,...,Wn}构成n棵二叉树的初始集合F={T1,T2,T3,...,Ti,...,Tn},其中每棵二叉树Ti中只有一个权值为Wi的根结点,它的左右子树均为空。

2)在F中选取两棵根结点权值最小的树作为新构造的二叉树的左右子树,新二叉树的根结点的权值为其左右子树的根结点的权值之和。

3)从F中删除这两棵树,把这棵新的二叉树加入到集合F中。

4)重复2)、3)两步,直到集合F中只有一棵二叉树,便是哈夫曼树。

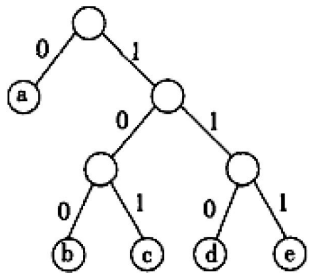
2.2 哈夫曼编码

哈夫曼编码,严格按照码字所对应符号出现概率的大小的逆序排列,得到的编码平均长度是最小的。(注:码字即为符号经哈夫曼编码后得到的编码,其长度是因符号出现的概率而不同,所以说哈夫曼编码是变长的编码。)根据构造好哈夫曼树,沿着从根结点到叶结点(包含原信息)的路径,向左孩子前进编码为0,向右孩子前进编码为1,当然也可以反过来规定。

例如:对下面这串出现了五种字符的信息(40个字符长)进行编码:cabcedeacacdeddaaabaababaaabbacdebaceada

五种字符的出现次数分别:a－16,b－7,c－6,d－6,e－5。

构造出的哈夫曼树如下:



于是我们得到了此信息的编码表:

a－0 b－100 c－101 d－110 e－111

可以将例子中的信息编码为:

101 0 100 101 111 110 111 0 101 0 101

码长共1×16+3×(7+6+6+5)=88位,若用ASCII码表示上述信息则需要8×40=320。可见,哈夫曼编码实现了数据压缩,但是其解码时间为O(lavg),其中lavg为码字的平均长度;并且解码器需要知道哈夫曼编码树的结构,因而编码器必须为解码器保存或传输哈夫曼编码树。这对于小量数据的压缩而言,这是很大的开销。那么,应用哈夫曼编码时,如何降低哈夫曼编码树的存储空间?

3 范式哈夫曼编码

Faller[1973]提出的自适应哈夫曼编码技术,使得哈夫曼编码树的存储空间降为零,即在使用某种约定的情况下,解码器能动态地重构出和编码器同步的哈夫曼编码树,而不需要任何附加数据。这样做的代价便是时间开销的增大。另一种技术是编码器和解码器使用事先约定的编码树,这种方法只能针对特定数据使用,不具备通用性。另外一种,也是最为常用的方法,便是范式哈夫曼编码。现在流行的很多压缩方法都使用了范式哈夫曼编码技术,如GZIB、ZLIB、PNG、JPEG、MPEG等。

3.1 范式哈夫曼编码概念

范式哈夫曼编码最早由Schwartz[1964]提出,它是哈夫曼编码的一个子集。其中心思想是:使用某些强制的约定,仅通过很少的数据便能重构出哈夫曼编码树的结构。其中一种很重要的约定是数字序列属性(numerical sequence property),它要求相同长度的码字是连续整数的二进制描述。例如,假设码字长度为4的最小值为0010,那么其它长度为4的码字必为0011,0100,0101,...;另一个约定:为了尽可能的利用编码空间,长度为i第一个码字f(i)能从长度为i-1的最后一个码字得出,即:f(i)=2(f(i-1)+1)。假定长度为4的最后一个码字为1001,那么长度为5的第一个码字便为10100。最后一个约定:码字长度最小的第一个编码从0开始。通过上述约定,解码器能根据每个码字的长度恢复出整棵哈夫曼编码树的结构。

3.2 范式哈夫曼编码码字构造

假设有如下的码长序列:

符号:a b c d e f g h i j k...u

码长:3 4 4 4 4 4 4 4 4 5 5...5

使用count[i]表示长度为i的码字的数目,first[i]表示长度为i的第一个码字的整数值。根据约定3,即first[3]=0可得到符号a的范式哈夫曼编码为000。再根据约定2,可得到first[4]=2×(first[3]+1)=2,进一步可知b的编码为0010。由约定1可构造出符号c的编码为0011,由此类推可构造出整个码字空间如下:

a=000(0);f=0110(6);k=10101(21);

b=0010(2);g=0111(7);

c=0011(3);h=1000(8);u=11111(31);

d=0100(4);i=1001(9);

e=0101(5);j=10100(20);

其中first[3]=0,first[4]=0010b=2,first[5]=10100b=20

3.3 范式哈夫曼解码

范式哈夫曼编码的一个很重要的特性:长度为i的码字的前j位的数值大于长度为j的码字的数值,其中i>j。如上例中的最小五位码10100,它的前四位1010大于任

(下转31页)

表1 索引表表结构

字段	类型	长度	备注
DATAID	GUID	16B	主键
REFCOUNT	NUMBER	4B	引用计数
FILENAME	TEXT	255b	文件名或文件路径

中心框架提供存取接口(API)。

3.2.1 索引表设计

索引表采用ACCESS数据库,表结构设计如表1所示。

表1说明如下:①如果数据存为文件,则FILENAME字段为文件路径;如果数据存在大文件里,则FILENAME字段的格式为FILENAME:②OFFSET(FILENAME为大文件名,OFFSET为数据在大文件中的偏移量),读数据时,根据FILENAME字段确定是否为文件;③由于经常性地增加和删除数据,而Access并不能有效地释放已分配的但被删除的记录空间,所以,需要定期压缩数据库,以迫使Access真正删除并回收这些空间,从而使得数据库尽量小,从而效率更高;

3.2.2 大文件设计

大文件的设计包括空间分配和空闲块管理两个方面,要考虑的问题有:

(1)设计合理的阈值。当一条数据的大小大于某个设定的阈值时,就单独存为一个文件,而小于这个阈值时,就作为一个记录存在大文件中,大文件按照阈值分配数据块空间。如果阈值太大、太小,都会造成空间效率太低。

(2)阈值的可扩展性。目前大部分数据都在300B以下,所以可将阈值设为384B。但根据数据大小的变化,可分别设定多个阈值,如384B、512B、1024B、2048B等,根据数据大小所处的区间选择存储。

(3)空闲块管理。大文件的空闲块通过位图向量法来管理,该方法从内存中划出若干个字节,为每个文件存储设备建立一张位示图。图中每一位对应一个物理块,“1”状态表示相应块已占用,“0”状态表示该块空闲。其主要优点是它可以全部或部分保存在主存中,实现高速分配。假如位图的大小为512k,则可管理的空闲块的个数为512K*8,即512K*8*512byte(blocksize)=2T(byte)。

大文件的数据结构设计如下:

```
数据块:dataBlock
typedef struct
{   dataHead *head;
    void * data;
}
typedef dataHead
{   GUID id;
    Unsigned Long size;
    Unsigned long count;
}
```

3.2.3 磁盘文件设计

对于用户要求存为文件的数据,则直接存为磁盘文件,需要考虑的问题就是文件存放路径。如果把所有文件存在同一个目录下,则时间效率低,如何使大量文件平均分配在不同目录下?目前是根据GUID来解析,一级目录有256个,根据GUID除以0XFF的余数决定存放路径。

4 结语

本课题主要是对协作区模块数据存储的研究和改进。在数据存储方面,提出改变原来分层存储的模式,实施数据中心方案。协作区数据由数据中心统一管理,数据中心不属于协作区三层模型中的任意一层,独立为三层提供数据存取,这样减少了相同数据的重复存储,用户的磁盘空间占用量大大减少,减小到大约为原来的1/3。根据以往版本用户的反馈信息,和改进版本的测试数据进行对比,协作区的性能有了明显的改善。由于数据层、同步层更新数据占用大量系统资源,而导致应用层对用户操作反应迟缓的现象得到好转。

参考文献

[1] 罗杰文.Peer to Peer(P2P)综述.中科院计算技术研究所,2005-11-3.
[2] 陈姝,方滨兴,周勇林.P2P技术的研究与应用[J].计算机工程与应用,2002,18(13):20-24.
[3] GK-Star.http://www.dianji.com/
[4] Dejan S.Milojicic,Vana Kalogeraki,Zhichen Xu et al.P2P Computing.HP Laboratories Palo Alto HPL-2002-57,March 8th,2002;
[5] 汤庸,冀高峰.协同软件技术及应用[M].北京:机械工业出版社,2007.

(上接29页)

何的四位码。由这个特性,容易构造出范式哈夫曼编码的解码算法:

```
extern KBitInputStream bs;
int len=1;
int code=bs.ReadBit();
while(code>=first[len])
{
  code<<=1;
  code &=(bs.ReadBit());//append next input bit to code
  len++;
}
len--;
```

其中while循环用于确定码长,这是解码算法中至关重要的一步,确定码长的算法效率影响着整个解码算法的效率。比如我们要解码100110100序列,当循环至len=4的时候,code等于1001,大于len[4],因而循环继续,继续读取下一位,code=10011,len=5,小于len[5]=10100,所以循环结束,执行下面的len--代码,得到了正确的码字长度4。

算法实现需要注意:一定要使用code>=first[len],而不是code>first[len];另外,len--不能少。代码中index[len]表示长度为len的第一个码字的索引,index[3]=0,index[4]=1,index[5]=9。不难发现:

index[i]=count[i-1]+count[i-2]+...+count[1]+count[0]

其中count[0]=0。

4 结语

本文针对哈夫曼编码在实际应用中存在的不足:编码器必须为解码器保存或传输哈夫曼编码树,开销很大,如何降低哈夫曼编码树的存储空间这一问题。主要介绍一种常用的方法——范式哈夫曼编码以及解码实现算法,有效解决了哈夫曼编码存在的不足,节约了时间空间开销并在实际中得到广泛的应用。

参考文献

[1] Faller,N.1973.An Adaptive System for Data Compression.Record of the 7th Asilomar Conf.on Circuits,Systems and Computers(Pacific Grove,Ca.,Nov.),593-597.
[2] Schwartz E.S.Kallick B,.Generating a cannonical prefix encoding,Communications of the ACM 7(1964),166-169.
[3] 严蔚敏,吴伟民编著数据结构[M].清华大学出版社,2007.
[5] 王群芳,哈夫曼编码的另一种实现算法[J].安徽教育学院学报.