并行表压缩算法

黄竞伟　　戴大为

(武汉水利电力学院，430072)　　(武汉大学，430072)

PARALLEL TABLE COMPRESSION ALGORITHM

Huang Jingwei

(Wuhan University of Hydraulic and Electric Engineering, 430072)

Dai Dawei

(Wuhan University, 430072)

ABSTRACT

In this paper, we propose a parallel table compression algorithm using a WRAM of $n^{1-\varepsilon}$ processors. The time complexity of the algorithm is $O\left(mn^{\varepsilon}+\frac{s^2}{n^{1-\varepsilon}}\right)$. Our algorithm achieves linear speedup.

摘　要

本文给出了在具有$n^{1-\varepsilon}$ 台处理器的WRAM 机器上实现的并行表压缩算法. 其时间复杂性为$O\left(mn^{\varepsilon}+\frac{s^2}{n^{1-\varepsilon}}\right)$. 这个算法达到了线性加速.

§1. 引　　言

查找问题是计算机科学中经常遇到的一个问题，D. E. Knuth 在[1] 中描述了一种在查找中常用的称为trie 的数据结构；R. E. Tarjan 在[2] 中给出了Knuth 所描述的trie 结构的一种变形. 在下面的讨论中，我们采用Tarjan 所描述的trie 结构.

给定一个n 个整数的集合$F=\{a_1,a_2,\cdots,a_n\}$, 其中$0\le a_i\le N-1$, 这里N 为正整数，表示F 的一个静态trie 结构是一棵多叉树，树的每个结点有一个数据场存储F 中的元素，

1989 年12 月6 日收到，1990 年5 月15 日定稿.

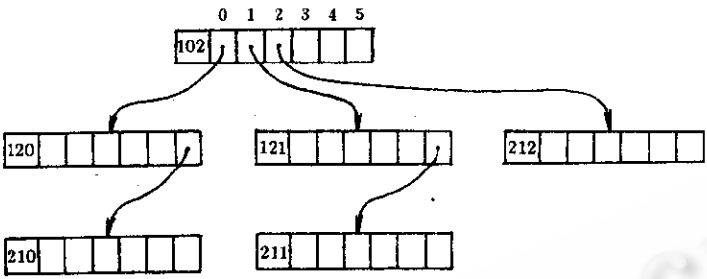

图 1

有一个具有n 个分量的指针数组指向它的子结点. 例如表示集合$F = \{102, 120, 121, 210, 211, 212\}$ 的一个静态trie 结构, 如图1 所示. 具体的trie 结构搜索算法参看[2].

在上述trie 结构中, 存储一个具有n 个整数的集合F 的存储花费为$O(n^2)$, 但在其n^2 个指针中只有$n-1$ 个非空. Tarjan 在[2] 中给出了一个将存储空间减少到$O(n \log\log n)$, 但不增加其查询花费的串行算法, 其具体做法如下: 将整个trie 结构结点中的指针数组看作一个n 阶矩阵, 每个结点中的指针数组看作此n 阶矩阵的一个行向量. 例如图1 中的trie 结构相应于下列矩阵:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

矩阵中的1 表示非空指针, 0 表示空指针. 减少trie 结构存储花费的办法是将trie 结构所对应的矩阵存放到一个一维数组中, 使得在保留trie 结构信息的条件下尽量减少存储花费.

问题的一般提法如下:

设A 是一个$m \times n$ 的(0, 1) 矩阵, 其中有s 个元素为1. 我们希望找到row displacement rd [i], $0 \le i \le m-1$, 使得:

$$\text{若} A[i,j] = 1 = A[i',j'] \text{且} (i,j) \ne (i',j'), \text{则有}$$
$$rd[i] + j \ne rd[i'] + j',$$

即我们要把一个二维数组A 的元素压缩到一个占有较少存储单元的一维数组C 中, 使A 中的第i 行的元素从C 的第$rd[i]$ 个位置放起, 则没有任何两个1 发生碰撞. 解决上述问题的算法称为表压缩算法.

表压缩算法在计算机科学领域里有丰富的应用, 除了能够用来减少一个trie 结构的存储空间外, 还能够用来存储LR parsing table[3], 以及其它的应用[4]. Tarjan 在[2] 中对串行的表压缩算法给出了详尽的分析. 本文的目的在于给出并行的表压缩算法及其分析.

§2. 计算模型

我们所用的计算模型是一种共享存储器的SIMD 机器WRAM[5], 它由若干个处理器

组成，每个处理器是一个RAM 机器，它们受中央控制器发出的单指令流的控制，并通过共享存储器的方式连接起来，当两个处理器交换数据时，它们通过共享存储器来实现. WRAM 允许多个处理器同时读或写一个共享存储单元. 关于处理写冲突的方法，本文的算法没有特别的要求，只要求有一个处理器写进去即可. 模型如图2 所示.

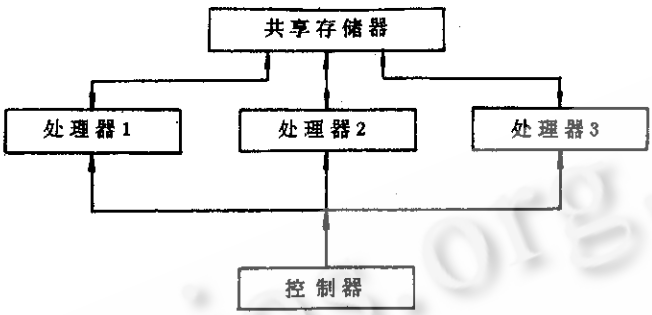

图 2

§3. 并行表压缩算法

Tarjan 在[2] 中给出了串行表压缩算法的一个$O(s^2+m)$ 时间的实现，但他假定算法的输入为非零元素的一个链接表. 而在我们下面的算法中，假定算法的输入为矩阵A, 此时Tarjan 的算法的时间复杂性为$O(s^2+mn)$.

Tarjan 在[2] 中所用串行表压缩算法的思想如下：

1. 按照矩阵A 行中1 的个数将行按照递减的次序重新排列.

2. 首先令$rd[0]=0$, 对$i\geq 1$ 假定已经选定了$rd[i-1]$, 我们这样选择$rd[i]$, 使$rd[i]$ 尽可能地小，且第i 行中的1 与前面已经放置好了的第$0, 1, \cdots, i-1$ 行中的1 没有碰撞.

在下面的并行表压缩算法中，为了实现上述的步骤1, 我们并不真正地将A 中的行按照1 的个数递减的次序排序，而是用一个数组Rank 将A 中行的次序按照1 的个数记录下来，Rank [0] 放1 的个数最多的行标，Rank [1] 放1 的个数第二多的行标，$\cdots$, Rank $[m-1]$ 放1 的个数最少的行标，在具体的实现过程中，即并行表压缩算法中的步骤一，二，三中，我们先用一个$m\times(n+1)$ 的数组L 按照行中1 的个数的多少记录A 的行标，用L 的第j 列记录A 中1 的个数为j 的行的行标$(j=0,1,\cdots,n)$, 并用数组Same 记录行标的个数，Sam $[j]$ 是L 中第j 列记录好的行标的个数. 然后再将L 中所记录的A 中的行标按照所含1 的个数递减的次序放到数组Rank 中. 为了并行地实现串行算法中的步骤2, 即在并行表压缩算法的步骤五中，在求放置第i 行的起始位置$rd[i]$ 时，先将第i 行中非零元素的列标放在数组List 中，并用变量length1 记录非零元素的个数. 由下面的定理1 知$0\leq rd[i]\leq s$, $i=0,1,\cdots,m-1$, 在检查$rd[i]$ 可能出现在哪一个位置时，我们使第j 个处理器检查第$(j-1)\frac{s}{n^{1-\varepsilon}}$ 至$\frac{s}{n^{1-\varepsilon}}-1$ 个位置，并用temprd $[i]$ 记录第j 个处理器检查的结果. 与串行表压缩算法不一样，我们选择的$rd[i]$ 并不一定是最小的.

先将后面要用到的一个定理写出如下：

定理1[2]：设矩阵A 有下列'harmonic decay' 性质：

$$H:\quad \text{对任 } l\geq 0, s(l+1)\leq \frac{s}{l+1},$$

则A 的每一个row displacement rd [i], $0 \le i \le m-1$, 满足$0 < rd[i] < s$, 其中$s(l)$ 为A 中具有大于l 个1 的行中1 的总数.

在下面的并行表压缩算法中, 假定矩阵A 满足'harmonic decay' 性质.

首先给出算法所需要的三个过程.

下面这个过程用$\left\lceil \frac{n}{2} \right\rceil$ 台处理器求n 个数的和$\sum_{i=1}^{n} B[i]$, 其结果放在$B[1]$ 中.

```
                    procedure sum (B, n);
1.  k: =⌈log n⌉;
2.  for i:=1 to n-2^k do in parallel
3.    B [i]: =B [i]+B[2^k+i];
4.  for i:=k down to 1 do
5.  for j:=1 to 2^(i-1) do in parallel
6.      B [j]: =B [j]+B [2^(i-1)+j];
```

显然过程sum 需要$O(\log n)$ 时间.

第二个过程是用n 台处理器求n 个数$A[0], \cdots, A[n-1]$ 的Prefix 和.

```
                    procedure prefix1 (D, n);
1.  for i:=0 to ⌈log n⌉ − 1 do
2.    begin
3.      for j:= 2^i + 1 to n do in parallel
4.        New D [j−1] :=D [j−2^i − 1]+ D [j];
5.      for j:=2^i + 1 to n do in parallel
6.        D [j−1]: =New D [j−1];
7.    end;
```

过程Prefix 1 也需要$O(\log n)$ 时间.

第三个过程是用$n^{1-\varepsilon}(0 < \varepsilon < 1)$ 台处理器求n 个数$A[0], \cdots, A[n-1]$ 的Prefix 和.

```
                    procedure prefix 2 (D, n);
1.  divide D [0], ···, D [n−1] into n^(1−ε) subsequences of n^ε each and assign a subsequence to each
    processor P_j.
2.  for j:= 1 to n^(1−ε) do in parallel
    P_j find the sum of its subsequence sequentially and assign the sum to B [j−1];
3.  prefix1 (B, n^(1−ε));
4.  for j:= 1 to n^(1−ε) − 1 do in parallel
        D [jn^ε]: =B [j−1]+D [jn^ε];
5.  for j:= 1 to n^(1−ε) do in parallel
    P_j find prefix sum of its subsequence sequentially.
```

在过程prefix2 的步骤1 中, 每个处理器从共享存储器中读入参数n, 这需要$O(1)$ 时间, 然后每个处理器P_j 从常数时间计算分配在它内部子序列的起始位置$(j-1)n^{\varepsilon}$ 及$jn^{\varepsilon}-1$, 故步骤1 需要$O(1)$ 时间. 显然步骤2, 4, 5 分别需要$O(n^{\varepsilon})$, $O(1)$, $O(n^{\varepsilon})$ 的时间, 步骤3 需要$O(\log n^{1-\varepsilon})$ 的时间, 故过程prefix2 的时间复杂性为$O(n^{\varepsilon})$.

在下面的并行表压缩算法中, 算法的输入是一个$m \times n$ 的(0, 1) 矩阵A, 其中有s 个元素为1. 用来实现算法的是一个具有$n^{1-\varepsilon}(0 < \varepsilon < 1)$ 台处理器的WRAM 机器.

并行表压缩算法

procedure table-compression (A, m, n);

一. 1. divide same [0], ···, same [n−1] into $n^{1-\varepsilon}$ subsequences of n^{ε} elements each and assign a subsequence to each processor.

2. for j:= 1 to $n^{1-\varepsilon}$ do in parallel
for k:= $(j-1)\ n^{\varepsilon}$ to $jn^{\varepsilon}-1$ do
same [k]:= 0;

3. same [n] := 0;

二. for i:= 0 to m−1 do

1. divide A [i, 0], ···, A [i, n−1] into $n^{1-\varepsilon}$ subsequences of n^{ε} elements each and assign a subsequence to each processor.

2. for j:= 1 to $n^{1-\varepsilon}$ do in parallel
processor P_j find the sum of its associated subsequence sequentially and write the sum into T [j].

3. find the sum S1= $\sum_{t=1}^{n^{1-\varepsilon}}$ T [t] using procedure sum and write i into L [same [S1], S1].

4. same [S1]:= same [S1]+1;

三. 1. for j:=1 to $n^{1-\varepsilon}$ do in parallel
for k:=(j−1) n^{ε} to $jn^{\varepsilon}-1$ do
E [k]:= same [k];
E [n]:=same [n];

2. prefix2 (E, n), E [−1]:=−1;

3. for j:= 1 to $n^{1-\varepsilon}$ do in parallel
r [j]:=1;

4. for i:=1 to $n^{1-\varepsilon}$ do in parallel
for j:=(i−1) n^{ε} to $in^{\varepsilon}-1$ do
if same [j] ≠ 0
then for k:=0 to same [j]−1 do
begin
rank [E [(i−1) $n^{\varepsilon}-1$+r [i]]]: =L [k, j];
r [i]:=r[i]+1;
end;
t:=1;
if same [n] ≠ 0
then for k:=0 to same [n] − 1 do
begin
rank [E [n−1]+t]: =L [k, n]; t:=t+1;
end;

四. 1. divide C [0], ···, C [s+n−1] into $n^{1-\varepsilon}$ subsequences of $\frac{s}{n^{1-\varepsilon}}+n^{\varepsilon}$ elements each and assign a subsequence to each processor.

2. for j:=1 to $n^{1-\varepsilon}$ do in parallel

```
    for k:=(j-1) [s/n^(1-ε) + n^ε] to j [s/n^(1-ε) + n^ε] - 1 do
      C [k]:=0;
    C [s+n]: =0;
  3. for i:=0 to m-1 do rd [i]:=0;
五. for i:=0 to m-1 do
  1. divide A [rank [i], 0], ···, A [rank [i], n-1] into n^(1-ε)
   subsequences of n^ε elements each and assign a
   subsequence to each processor.
  2. each processor find the sum of its associated
   subsequence sequentially and write them into length [j-1].
  3. prefix 1 (length, n^(1-ε)); length [-1] := 1;
   length1: =length [n^(1-ε) - 1];
  4. for j:=1 to n^(1-ε) do in parallel
    begin
     T [j]:=0;
     for k:=(j-1) n^ε to jn^ε - 1 do
       if A [rank [i], k]=1
          then begin
             T [j]: =T [j]+1; List [length [j-2]+T [j]]: =k;
               end;
    end;
  5. for j:=1 to n^(1-ε) do in parallel
    begin
      temprd [j]:=(j-1) s/n^(1-ε); K [j]:=1;
      while K [j] ≤ length1 and temprd [j] < j s/n^(1-ε) do
          begin
            if C [ temprd [j] + List [K [j]]]=1
                then begin
                  K [j]:=1; temprd [j]: =temprd [j]+1;
                  end
                else K [j]: =K [j]+1;
          end;
        if K [j]=length 1 + 1
          then rd [rank [i]]: =temprd [j];
    end;
  6. for k:=1 to length 1 do
      C [ rd [rank [i]]+List [k]]:=1;
```

定理2: 若矩阵A 具有'harmonic decay' 性质，则并行表压缩算法需要$O\left(mn^{\varepsilon}+\frac{s^2}{n^{1-\varepsilon}}\right)$的时间计算$A$ 的row displacement rd [i], $0 \le i \le m-1$.

证明：步骤一需要$O(n^{\varepsilon})$ 的时间. 在步骤二中，1, 4 需要$O(1)$ 的时间，2 需要$O(n^{\varepsilon})$的时间，3 需要$O(\log n^{1-\varepsilon})$ 的时间，故步骤二共需要$O(mn^{\varepsilon})$ 的时间. 在步骤三中，1, 2需要$O(n^{\varepsilon})$ 的时间，3 需要$O(1)$ 的时间，4 需要$O(m+n^{\varepsilon})$ 的时间，故步骤三需要$O(m+n^{\varepsilon})$的时间. 步骤四需要$O\left(\frac{s}{n^{1-\varepsilon}}+n^{\varepsilon}+m\right)$ 的时间. 在步骤五中，1 需要$O(1)$ 的时间，2,

4 需要$O(n^{\varepsilon})$ 的时间，3 需要$O(\log n^{1-\varepsilon})$ 的时间，5 需要$O\left(\frac{s}{n^{1-\varepsilon}}l_i\right)$ 的时间，其中l_i 为A 的第i 行所包含1 的个数，6 需要$O(l_i)$ 的时间，故步骤五共需要$O\left(mn^{\varepsilon}+\sum_{i=1}^{m-1}\frac{s}{n^{1-\varepsilon}}l_i\right)$ $=O\left(mn^{\varepsilon}+\frac{s^2}{n^{1-\varepsilon}}\right)$ 的时间. 所以算法共需要$O\left(mn^{\varepsilon}+\frac{s^2}{n^{1-\varepsilon}}\right)$ 的时间.

§4. 进一步研究的若干问题

在本文中，我们给出了用WRAM 机器实现的并行表压缩算法，在算法中，我们假定矩阵A 满足'harmonic decay' 性质，若矩阵A 不满足'harmonic decay' 性质，则我们可以从矩阵A 构造出一个满足'harmonic decay' 性质的矩阵B, 在将上述的并行表压缩算法应用到B 后，则可以得到矩阵A 到一维数组C 的一个压缩，怎样从A 构造B, 使B 满足'harmonic decay' 性质的并行算法将另文讨论. 另一个值得研究的问题是怎样在PRAM 或PRAC 机器上实现并行的表压缩算法，且使效果和在WRAM 机器上实现的效果相近，这值得我们进一步研究.

参考文献

[1] D. E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.

[2] R. E. Tarjan and C.-C. A. Yao, Storing a Sparse Table, CACM, 22: 11 (1979), 606–611.

[3] A. V. Aho and J. D. Ullman, Principles of Compiler Design, Addison-wesley, 1977.

[4] R. E. Tarjan, Graph Theory and Gaussian Elimination, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, Eds., Academic Press, New York, 1976, 3–22.

[5] A. Borodin and J. E. Hopcroft, Routing, Merging and Sorting on Parallel Models of Computation, J. Comput. & Syst. Sci., 30 (1985), 130–145.

'91 国际软件工程研讨会在北京举行

1991 年10 月28 日至30 日，'91 国际软件工程研讨会在北京大学举行. 参加会议的有来自日本、美国和德国的专家学者22 人及我国软件专家42 人. 有关单位的领导张效祥、陈佳洱、姜均露、杨天行、陈冲、陈树楷等同志出席了会议.

会上报告了计算机软件工程方面的学术论文近20 篇，参观了我国“七·五”攻关成果“集成化软件工程支撑环境JB 系统”的演示. 中外专家对当前软件工程方面的技术发展趋势、遇到的问题及可能采取的解决途径进行了热烈的讨论，交流了各自的研究成果和心得体会. 特别是对软件过程、软件方法学、面向对象技术、软件工程环境及工具、软件工程与人工智能的关系、软件重用等方面展开了深入的探讨. 整个会议思想活跃、气氛热烈. 与会代表表示满意，并希望经常举行这样有益的学术交流活动.

张效祥	中国计算机学会理事长	杨天行	机电部计算机司司长
陈佳洱	北京大学副校长	陈　冲	机电部计算机司软件处处长
姜均露	国家计委科技司副司长	陈树楷	中国计算机学会秘书长