基于PRAM模型的二叉树A序列并行算法的研究

孙玉强[1,2] 顾玉宛[1] 张聪品[2] 张英丽[1]
(江苏工业学院信息科学与工程学院 常州 213164)[1]
(河南师范大学计算机与信息技术学院 新乡 453007)[2]

摘 要 运用并行计算的PRAM模型研究二叉树A序列问题,提出了二叉树的A序列的一种并行算法,并以应用实例对并行算法的过程进行详细描述和验证性分析。二叉树A序列的并行算法,为应用到二叉树序列遍历的系统与应用程序的并行化问题的解决提供借鉴和参考。

关键词 A序列,二叉树,并行算法,PRAM模型

Research on Parallel Algorithm of A-order in Binary Tree Based on PRAM Model

SUN Yu-qiang[1,2] GU Yu-wan[1] ZHANG Cong-pin[2] ZHANG Ying-li[1]
(School of Information Science & Engineering, Jiangsu Polytechnic University, Changzhou 213164, China)[1]
(College of Computer & Information Technology, Henan Normal University, Xinxiang 453007, China)[2]

Abstract The problem of A-order of binary tree was studied with the PRAM model of parallel computation and a parallel algorithm for A-order of binary tree was proposed. The process of the parallel algorithm was described and analysed with an application instance. The parallel algorithm of A-order of binary tree provides using and reference for application in the system of binary tree sequence traverse and solving the parallelism problem of application program.

Keywords A-order sequence, Binary tree, Parallel algorithm, PRAM model

1 引言

A序列是二叉树结构的重要参数,对树的各种操作及算法的影响较大[1,2]。二叉树是一种非线性结构,每个结点都可能有两棵子树,其遍历算法较为复杂[3]。Hayedeh Ahrabian等提出一种二叉树生成并行算法,A序列是算法中主要数据结构[4]。PRAM(parallel random access machine)模型使并行算法的设计者可以把处理器的能力看成是无限的,就像程序员有了虚拟内存后可以把内存看成无限的一样。一个PRAM由一个控制单元、全局内存和一组处理器集合组成,每个处理器有它自己的私有内存。所有处理器执行相同的指令,但每个处理器处理的数据不一样。本文运用并行计算的PRAM模型研究二叉树的A序列问题,提出一种基于PRAM模型的二叉树的A序列的并行算法,并与先序遍历并行算法进行比较分析。算法中采用"并行读排他写"PRAM模型,即CREW(Concurrent Read Exclusive Write)方式。在这种模型中,并行读是允许的,即在同一个指令步骤中,多个处理器可以读取相同的内存。但是不允许出现写冲突。本文后半部分以应用实例对该并行算法的执行过程进行详细描述和结果验证性分析。

2 先序遍历并行算法

整个算法可以分为4步:

第一步 构造一个单链表。

为了描述的直观、方便,我们可以将无向图改造成一个相对应的有向图,改造方法是将无向图的每条无向边改造成一条向下的有向边,并添加结点,分3种情况:①若该结点没有左子树则添加左子树;②若该结点没有右子树则添加右子树;③若该结点既没有左子树又没有右子树则添加左子树和右子树;最终使得原二叉树的结点均有左右子树。图1(a)所示的无向图可以改造成相应的有向图1(b)。

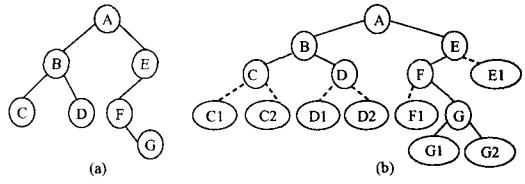

图1 原二叉树及改造后的二叉树

算法的实现使用了一个特殊的数据结构来表示二叉树。对改造后的二叉树的每个结点,数据结构保存结点的双亲、左孩子和右孩子,利用这3个参数来描述一个树结点在树中的相对位置。对如图1(b)所示的二叉树,其数据结构表示如表1所列。

构造单链表的过程就是在改造后的有向图中寻找后继边的过程。按照先序遍历算法的思想,所有的处理器同时处理分配给它的一条边的后继边的问题,最终得到由全部有向边构成的一个单链表,单链表的每个元素对应于改造后的有向图中的一条有向边。

到稿日期:2008-05-18 本文受江苏省高校自然科学基金(项目编号:06KJB520022)和河南省高校自然科学基金(项目编号:200510476016)资助。
孙玉强 教授,博士;顾玉宛 硕士研究生;张聪品 副教授;张英丽 硕士。

表 1　图 1(b)中的二叉树的数据结构

	A	B	C	C1	C2	D	D1	D2	E	E1	F	F1	G	G1	G2
Parent	Null	A	B	C	C	B	D	D	A	E	E	F	F	G	G
Lchild	B	C	C1	Null	Null	D1	Null	Null	F	Null	F1	Null	G1	Null	Null
Rchild	E	D	C2	Null	Null	D2	Null	Null	E1	Null	G	Null	G2	Null	Null

具体的某个处理器寻找后继边的过程可以这样描述，设处理器为 p_m，分配给它处理的有向边是(i,j)(即该边从结点 i 指向结点 j)，则寻找有向边(i,j)的后继边的问题可以根据有向边(i,j)的不同类型分别处理。

Parent [j]=i，说明有向边(i,j)是从双亲结点到它的一个孩子结点的向下边。同理，从数据结构的概念中可以得出这样的结论，一条向下的边(i,j)在二叉树中的相对位置可以有如下 3 种情况。

第一种情况是结点 j 有左孩子结点，则根据先序遍历的思想可得，边(i,j)的后继边是从结点 j 指向结点 j 的左孩子结点 Lchild 构成的边。

第二种情况是结点 j 无左孩子结点有右孩子结点，则根据先序遍历的思想可得，边(i,j)的后继边是从结点 j 指向结点 j 的右孩子结点 Rchild 构成的边，并将结点 j 的右孩子置为空。

第三种情况是结点 j 既无左孩子结点又无右孩子结点，则根据先序遍历的思想可得，后退 j 结点到 j 结点的双亲结点，然后再判断该结点是否存在右孩子结点，若有则根据先序遍历的思想可得，边(i,j)的后继边是从结点 j 指向结点 j 的右孩子结点 Rchild 构成的边，并将结点 j 的右孩子置为空；若无则继续回退。

第二步　给单链表中的每个元素赋权值。

所有的处理器对分配给它的元素赋权值(这里的元素是指第一步形成的单链表的元素)。一条向下遍历的边(i,j)，若 j=Lchild [i]，将单链表中对应的边的元素赋权值为 1；若 j=Rchild [i]，将单链表中对应的边的元素赋权值为 0。

第三步　计算单链表中元素的位序。

单链表中每个元素需要分配一个处理器去计算其位序。一棵有 N 个结点的二叉树，经改造后有 $2N+1$ 个结点的二叉树，有 $2N$ 条向下的有向边，这意味着单链表中有 $2N$ 个元素。所以，需要 $2N$ 个处理器来计算单链表中元素的位序。利用计算单链表位序的后缀和算法可以算出单链表中每个元素的位序。

第四步　挑出权值为 1 的元素，求相应结点的先序遍历顺序号。

由于二叉树的改造原则，权值为 1 的元素所代表的向下边(i,j)，结点 i 在边(i,j)被遍历时均被访问到，因此，用单链表中权值为 1 的元素的位序表示它所代表的边(i,j)中结点 i 的位序，从而可以得到原二叉树中每个全部结点的位序。

结点的位序是结点被访问的先后顺序的逆序。只要用原二叉树的结点个数 $N+1$ 减去每个结点的位序，就可以得到每个结点的先序遍历顺序号。与前几步一样，这里也是一个处理器计算一个结点的顺序号，多个处理器并行工作。最后得到了一棵二叉树的先序遍历的结点顺序。

完整的算法描述如下：

```
PREORDER. TREE. TRVERSAL
Global n=2N+1 {N是原二叉树的结点数,n是改造后的二叉树的结点数}
  Parent [1…n] {父结点数组}
  Lchild [1…n] {左孩子结点数组}
  Rchild [1…n] {右孩子结点数组}
  Succ [1…n,1…n] {后继边二维数组}
  Position [1…n,1…n] {链表元素的位序数组}
  Preorder [1…n] {结点顺序号数组}
Begin
{P(i,j)是处理对应边(i,j)的一个处理器}
{(i,j)是一条边,1≤i≤n,1≤j≤n}
Spawn (set of all P(i,j))
  For all P(i,j) do
    {构造一个单链表}
  If Parent [j]=i then
    If Lchild [j]! =Null then
      Succ [(i,j)]<—(j,Lchild [j])
    Else
     {If Rchild [j]! =Null then
      { Succ [(i,j)]<—(j,Rchild [j]);
        Rchild [i] <— Null
        }
      Else
        {1°If Rchild [i]! =Null then
         {Succ [(i,j)]<—(i,Rchild [i]);
          Rchild [i] <— Null
          }
          Else i=Parent [i]
         Goto 1°}
      Endif
        }
    Endif
  Endif
  {给单链表中的每个元素赋权值}
  If j=Lchild [i] then
    Position [(i,j)] <—1
  Elseif j=Rchild [i] then
  Position [(i,j)] <—0
  Endif
  {计算单链表中元素的位序}
  For k <—1 to [log(n—1)] do
      Position [(i,j)] <—Position [(i,j)] +Position [Succ [(i,
j)]]
      Succ [(i,j)] <—Succ [Succ (i,j)]
  Endfor
  {求结点的先根遍历顺序号}
  If j=Parent [i] then
      Preorder[i]<—N+1- Position [(i,j)]
  Endif
 Endfor
End
```

3　A 序列的并行算法

A 序列的定义[5]：一棵有 n 个结点的二叉树 T，T 的 A 序

(下转第 294 页)

是对全局本体里的 Material 类进行相同概念的推理，其中谓语是 equivalentClass，主语是 Material，推理得到的结果放置在容器中，使用迭代器 i 就可以将结果顺序取出。

结束语 语义网格在网格迅速发展和普及的未来将起着越来越重要的作用。由于它可以理解用户的问题并且提供解决方案，它将会为终端用户提供更好更先进的服务。本文的研究对推动结构工程研究领域的信息集成与共享有着积极的意义。

当前的工作还处在初级的阶段，还有许多关键的问题需要进一步解决，如本体查询性能问题，目前采用的本体查询语言 SPARQL，其性能耗费远高于一般查询语言（如 SQL 和 XQuery），且当连接 SPARQL 和 XQUERY，或者连接 SPARQL 与 SQL 时非常复杂且花费巨大，这些都需要进一步研究、实践。

参 考 文 献

[1] Berners-Lee T, Hendler J, Lassila O. The Semantic Web. Scientific American, May 2001

[2] Trinh Q, Barker K, lhajj R. Semantic Interoperability between Relational Database Systems // 11th International Database Engineernig and Applications Symposium (IDEAS 2007). 200 IEEE

[3] Semantic Grid Community Portal. http://www.semanticgrid.org, 2004, 12

[4] 洪晓伟. 基于 XML 异构数据集成的研究[D]. 东南大学图书馆, 2004

[5] Goble C, Roure D D. The Semantic Grid: Myth Busting and Bridge Building. http://www.semanticgrid.org/docs/ECAISemanticGrid/ECAISe-manticGridFinal.pdf, 2004, 12

[6] 邓志鸿，唐世渭，张铭，等. Ontology 研究综述[J]. 北京大学学报：自然科学版，2002，38(5)：730-738

[7] Isabel F, Cruz Huiyong Xiao, Feihong Hsu. An ontology-based framework for XML semantic integration [C] // Washington: IDEAS'04 Workshop, Proceedings of the International Database Engineering and Applications Symposium IEEE computer Society. 2004: 217-226

(上接第 257 页)

列：$A=(a_1,a_2,\cdots,a_n)$，$a_i(1\leqslant i\leqslant n)$表示第 i 个结点的左子树上的结点数目，$a_1,a_2,\cdots,a_n$ 是按二叉树 T 的先序来排列的。

用上述相同的方法构造图 1(a)所示的二叉树，其数据结构表示如表 2 所列。

表 2 图 1(a)中的二叉树的数据结构

	A	B	C	D	E	F	G
Parent	Null	A	B	B	A	E	F
Lchild	B	C	Null	Null	F	Null	Null
Rchild	E	D	Null	Null	Null	G	Null

算法如下：

```
For all pi par-do
  r=0;j<—i
If Lchild [j]! = Null then
  {r=r+1;j=Lchild [j]}
  Elseif j! =i then
  {If Rchild [j]! = Null then
    r=r+1;
  Endif
    j=Parent [j]
  }
 Endif
Endfor
```

4 实例分析

以图 1(a)所示的树为例，给出本文算法的实现过程。

第一步构造单链表，结果如图 2 所示。

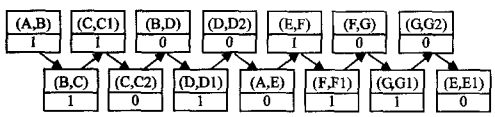

图 2 单链表示意图

第二步给单链表中的每个元素赋权值，结果如图 2 所示。

第三步计算单链表的每个元素的位序，每步计算所得结果如表 3 所列。

表 3 位序计算结果列表

AB	BC	CC1	CC2	BD	DD1	DD2	AE	EF	FF1	FG	GG1	GG2	EE1	
1	1	1	0	0	1	0	0	1	1	0	1	0	0	
2	2	1	0	1	1	0	1	2	1	1	1	0	0	1 跳
3	2	2	1	1	2	2	2	3	2	1	1	0	0	2 跳
4	4	4	3	4	4	3	3	3	2	1	1	0	0	4 跳
7	6	5	4	4	4	3	3	3	2	1	1	0	0	8 跳

第四步求结点的先序遍历顺序号权值为 1 的边有$(A,B),(B,C),(C,C1),(D,D1),(E,F),(F,F1),(G,G1)$，从而可以得到结点 A,B,C,D,E,F,G 的位序分别为 7,6,5,4,3,2,1，由于结点个数 $N+1=8$，可以根据各结点的位序计算出各结点的先序遍历顺序号依次为 1,2,3,4,5,6,7，即这棵二叉树的遍历顺序为 A,B,C,D,E,F,G。

最后，根据第 3 节中 A 序列的并行算法可求得 $A=(3,1,0,0,2,0,0)$。

结束语 一棵有 N 个结点的二叉树，经改造后有 $2N+1$ 个结点的二叉树，有 $2N$ 条向下的有向边，因此，该算法需要 $2N$ 个处理机。在该算法中，求单链表中元素位序的计算时间复杂度为 $O(\log 2N)$，算法其余部分的计算时间是常数，所以该算法的时间复杂度为 $O(\log 2N)$。

参 考 文 献

[1] Ahrabian H, Nowzari - Dalini A. Parallel generation of binary trees in A-order. Parallel Computing, 2005, 31: 948-955

[2] Ahrabian H, Nowzari - Dalini A. On the generation of binary trees in A-order. Int. J. Comput. Math., 1999, 71: 1-7

[3] 严蔚敏，吴伟民. 数据结构(C 语言版). 北京：清华大学出版社，1999

[5] Ahrabian H, Nowzari - Dalini A. Parallel generation of binary trees in A-order. Parallel Computing, 2005, 31: 948-955

[4] Korsh J F. A-order generation of K-ary trees with 4K-4 letter alphabet. J. Infom. Optim. Sci., 1995, 16(3): 557-567