并行计算
Parallel Computing

主讲人　孙广中

Spring, 2022

第二篇 并行算法的设计

第五章 并行算法与并行计算模型
第六章 并行算法基本设计策略
第七章 并行算法常用设计技术
第八章 并行算法一般设计过程

第六章 并行算法基本设计策略

6.1 串行算法的直接并行化

6.1.1 设计方法描述

6.1.2 快排序算法的并行化

6.2 从问题描述开始设计并行算法

6.3 借用已有算法求解新问题

设计方法的描述

- 方法描述
  - 发掘和利用现有串行算法中的并行性，直接将串行算法改造为并行算法。
- 评注
  - 由串行算法直接并行化的方法是并行算法设计的最常用方法之一;
  - 不是所有的串行算法都可以直接并行化的;
  - 一个好的串行算法并不能并行化为一个好的并行算法;
  - 许多数值串行算法可以并行化为有效的数值并行算法。

第六章 并行算法基本设计策略

6.1 串行算法的直接并行化

6.1.1 设计方法描述

6.1.2 快排序算法的并行化

6.2 从问题描述开始设计并行算法

6.3 借用已有算法求解新问题

快排序算法的并行化（1）

- SISD上的快排序算法6.1

输入：无序序列（A_q……Ar)

输出：有序序列（A_q……Ar)

```
Procedure Quicrsort(A，q，r);
 Begin
   if q<r then
    (1) x=A_q
    (2) s=q
    (3) for i=q+1 to r do
          if A_i≤x then
             s=s+1
             swap(A_s,A_i)
          end if
    endfor
    (4) swap(A_q, A_s)
    (5) Quicksort(A, q, s)
    (6) Quicksort(A, s+1, r)
   end if
 end
```

对于长度为n的序列，在最坏情况下的划分的两个子序别为n-1及1的长度、相应的运行时间为$t(n)=t(n-1)+\Theta(n)$，其解为$t(n)=\Theta(n^2)$.

理想的情况是所划分的两个子序列等长，相应的运行时间为$t(n)=2t(n/2)+\Theta(n)$,其解为$t(n)=\Theta(n\log n)$.

快排序算法的并行化（2）

- 快排序的并行化
  - 一种自然的并行化方法是并行地调用快排序对两个所划分的子序列进行快排序。这种方法并不改变串行算法本身的属性，很容易改成并行形式。但是，如果用n个PE排序n个数，若用一个PE将$(A_1.....A_n)$划分成$(A_1.....A_s)$，$(A_{s+1}.....A_n)$是不会得到成本最优算法的，因为单是划分时就有$\Omega(n)$，所以$C(n)=p(n)t(n)=\Omega(n^2)$.
  - 可见，只有将划分步并行化，才有可能得到成本最优算法.
  - PRAM-CRCW上快排序算法
    - 构造一棵二叉排序树，其中主元是根
    - 小于等于主元的元素处于左子树，大于主元的元素处于右子树
    - 其左、右子树分别也为二叉排序树

快排序算法的并行化（3）

- 算法6.2 APRAM-CRCW上的快排序二叉树构造算法

输入：序列(A_1,…,A_n)和n个处理器

输出：供排序用的一棵二叉排序树

Begin

(1)for each processor i par-do

(1.1)root=i

(1.2)f_i=root

(1.3)LC_i=RC_i=n+1

end for

(2)repeat for each processor i<>root par-do

if (A_i<A_{fi})∨(A_i=A_{fi}∧i<f_i) then

(2.1)LC_{fi}=i

(2.2)if i=LC_{fi} then exit else f_i=LC_{fi} endif

else

(2.3)RC_{fi}=i

(2.4)if i=RC_{fi} then exit else f_i=RC_{fi} endif

endif

end repeat

End

注：(1)A_i，LC_i，RC_i，root是SM变量；f_i可以是LM变量；

(2)时间为O(logn)

第六章 并行算法基本设计策略

6.1 串行算法的直接并行化

6.2 从问题描述开始设计并行算法

6.2.1 设计方法描述

6.2.2 有向环k着色算法的并行化

6.3 借用已有算法求解新问题

设计方法的描述

- 方法描述
  - 从问题本身描述出发，不考虑相应的串行算法，设计一个全新的并行算法。
- 评注
  - 挖掘问题的固有特性与并行的关系；
  - 设计全新的并行算法是一个挑战性和创造性的工作；
  - 利用串的周期性的PRAM-CRCW算法是一个很好的范例；

第六章 并行算法基本设计策略

6.1 串行算法的直接并行化

6.2 从问题描述开始设计并行算法

6.2.1 设计方法描述

6.2.2 有向环k着色算法的并行化

6.3 借用已有算法求解新问题

有向环k着色算法的并行化（1）

- K着色问题

设有向环G=(V,E)，G的k着色问题：构造映射c:
V→{0,1,2,...,k-1}，使得如果<i, j> ∈E，则c[i]≠c[j]

- 3着色串行算法

从一顶点开始，依次给顶点交替用两种颜色着色，如果顶点数为奇数则需要第3种颜色。

注：该串行算法难以并行化。这时需要将顶点划分为若干类，每类指派相同颜色，最后再将分类数减为3。

有向环k着色算法的并行化（2）

- 基本并行k着色算法

算法: SIMD-EREW模型

//输入初始点着色c(i)，输出最终着色c'(i)

begin

for i=1 to n par-do

(1)令k是c(i)和c(i的后继)的最低的不同二进制位

(2)c'(i)= 2k+c(i)$_k$　　//c(i)$_k$为c(i)的二进制第k位

end for

end

注: (1)算法的正确性需要证明;

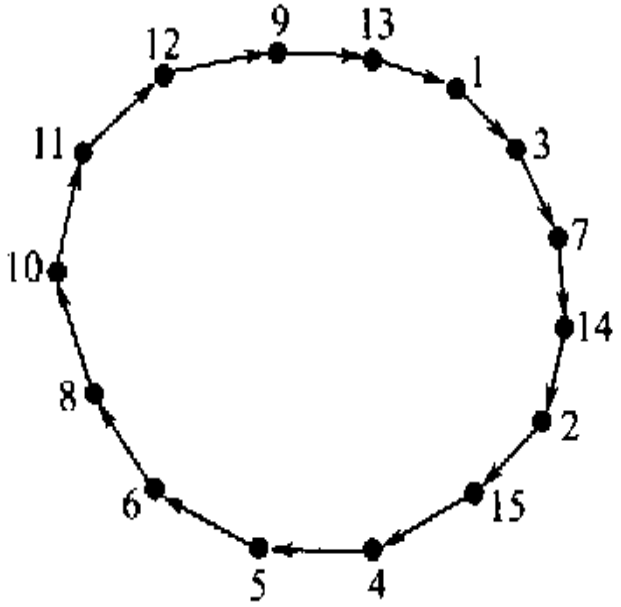
(2)算法的运行时间和工作量?

(3)算法应用的是破对称技术，算法中打破了顶点的对称性，并对分类进行了压缩;

(4)在此算法的基础上如何构造3着色算法?

有向环k着色算法的并行化（3）

- 图例



(a)

v	c	k	c'
1	0001	1	2
3	0011	2	4
7	0111	0	1
14	1110	2	5
2	0010	0	0
15	1111	0	1
4	0100	0	0
5	0101	0	1
6	0110	1	3
8	1000	1	2
10	1010	0	0
11	1011	0	1
12	1100	0	0
9	1001	2	4
13	1101	2	5

(b)

第六章 并行算法基本设计策略

6.1 串行算法的直接并行化

6.2 从问题描述开始设计并行算法

6.3 借用已有算法求解新问题

6.3.1 设计方法描述

6.3.2 利用矩阵乘法求所有点对间最短路径

设计方法的描述

- 方法描述
  - 找出求解问题和某个已解决问题之间的联系;
  - 改造或利用已知算法应用到求解问题上。
- 评注
  - 这是一项创造性的工作;
  - 使用矩阵乘法算法求解所有点对间最短路径是一个很好的范例。

第六章 并行算法基本设计策略

6.1 串行算法的直接并行化

6.2 从问题描述开始设计并行算法

6.3 借用已有算法求解新问题

6.3.1 设计方法描述

6.3.2 利用矩阵乘法求所有点对间最短路径

利用矩阵乘法求所有点对间最短路径（1）

- 计算原理

有向图G=(V,E)，边权矩阵$W=(w_{ij})_{n\times n}$，求最短路径长度矩阵$D=(d_{ij})_{n\times n}$，d_{ij}为v_i到v_j的最短路径长度。假定图中无负权有向回路，记$d^{(k)}_{ij}$为v_i到v_j至多有k-1个中间结点的最短路径长，$D^k=(d^{(k)}_{ij})_{n\times n}$，则

(1) $d^{(1)}_{ij}=w_{ij}$　　当 i<>j (如果v_i到v_j之间无边存在记为∞)

$d^{(1)}_{ij}=0$　　当 i=j

(2) 无负权回路 ➔ $d_{ij}=d^{(n-1)}_{ij}$

(3) 利用最优性原理：$d^{(k)}_{ij}=\min_{1\leqslant l\leqslant n}\{d^{(k/2)}_{il}+d^{(k/2)}_{lj}\}$

视：”+” → “×”，“min” → “∑”，则上式变为

$d^{(k)}_{ij}=\sum_{1\leqslant l\leqslant n}\{d^{(k/2)}_{il}\times d^{(k/2)}_{lj}\}$

(4) 应用矩阵乘法：$D^1\rightarrow D^2\rightarrow D^4\rightarrow \ldots\rightarrow D^{2^{\log n}}$ $(=D^n)$

- SIMD-CC上的并行算法：p183算法6.5

算法 6.5　SIMD-CC 上求所有点对间最短路径算法

输入：$A(0,j,k)=w_{jk}$，$0\leqslant j,k\leqslant n-1$

输出：$C(0,j,k)$中是 v_j 到 v_k 的最短路径长度，$0\leqslant j,k\leqslant n-1$

Begin

(1) /* 构筑矩阵 $\boldsymbol{D}^1$，并将其存入 A 和 B 寄存器中 */

for $j=0$ **to** $n-1$ **par-do**

for $k=0$ **to** $n-1$ **par-do**

(1.1) **if** $j\neq k$ & $A(0,j,k)=0$ **then**

$B(0,j,k)=\infty$

endif

else

(1.2) $B(0,j,k)=A(0,j,k)$

endfor

endfor

(2) /* 调用算法 9.6 构筑矩阵 $\boldsymbol{D}^2,\boldsymbol{D}^4,\cdots,\boldsymbol{D}^{n-1}$ */

for $i=1$ **to** $\lceil\log(n-1)\rceil$ **do**

(2.1) DNS MULTIPLICATION (A,B,C)　/* 调用算法 9.6 */

(2.2) **for** $j=0$ **to** $n-1$ **par-do**

for $k=0$ **to** $n-1$ **par-do**

(i) $A(0,j,k)=C(0,j,k)$

(ii) $B(0,j,k)=C(0,j,k)$

endfor

endfor

endfor

End

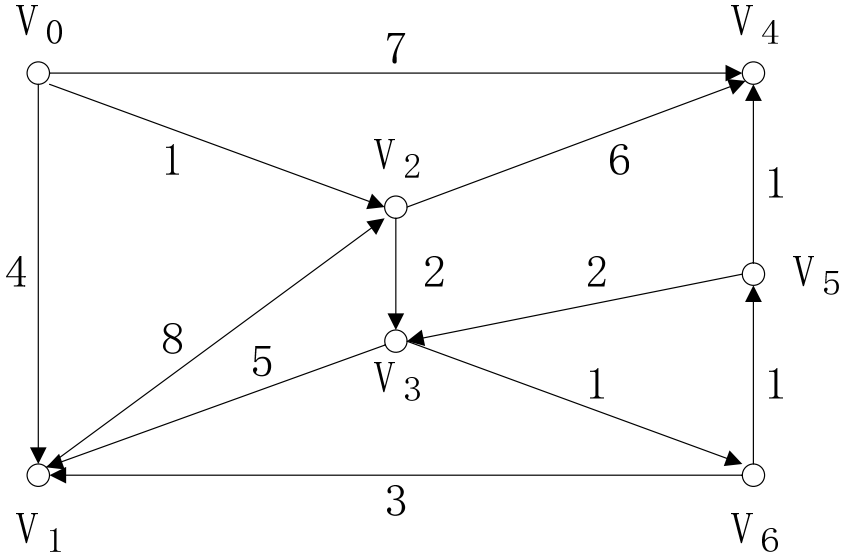
利用矩阵乘法求所有点对间最短路径（2）

- 时间分析

  每次矩阵乘时间O(logn), 共作$\lceil \log n \rceil$次乘法

  ==> $t(n)=O(\log^2 n)$, $p(n)=n^3$

- 计算示例



(a)

	0	1	2	3	4	5	6
0	0	4	1	0	7	0	0
1	0	0	8	0	0	0	0
2	0	0	0	2	6	0	0
3	0	5	0	0	0	0	1
4	0	0	0	0	0	0	0
5	0	0	0	2	1	0	0
6	0	3	0	0	0	1	0

(b)

利用矩阵乘法求所有点对间最短路径（3）

- 计算示例

(c)

	0	1	2	3	4	5	6
0	0	4	1	∞	7	∞	∞
1	∞	0	8	∞	∞	∞	∞
2	∞	∞	0	2	6	∞	∞
3	∞	5	∞	0	∞	∞	1
4	∞	∞	∞	∞	0	∞	∞
5	∞	∞	∞	2	1	0	∞
6	∞	3	∞	∞	∞	1	0

(d)

	0	1	2	3	4	5	6
0	0	4	1	3	7	∞	∞
1	∞	0	8	10	14	∞	∞
2	∞	7	0	2	6	∞	3
3	∞	4	13	0	∞	2	1
4	∞	∞	∞	∞	0	∞	∞
5	∞	7	∞	2	1	0	3
6	∞	3	11	3	2	1	0

(e)

	0	1	2	3	4	5	6
0	0	4	1	3	7	5	4
1	∞	0	8	10	14	12	11
2	∞	6	0	2	5	4	3
3	∞	4	12	0	3	2	1
4	∞	∞	∞	∞	0	∞	∞
5	0	6	14	2	1	0	3
6	∞	3	11	3	2	1	0

(f)

	0	1	2	3	4	5	6
0	0	4	1	3	6	5	4
1	∞	0	8	10	13	12	11
2	∞	6	0	2	5	4	3
3	∞	4	12	0	3	2	1
4	∞	∞	∞	∞	0	∞	∞
5	∞	6	14	2	1	0	3
6	∞	3	11	3	2	1	0

1. A是一个大小为n的布尔数组，欲求出最小的下标i且A[i]为真，试设计一个常数时间的PRAM-CRCW并行算法。如果使用PRAM-CREW模型，运行时间如何？

Hint:
```
1. copy A[1..n] to B[1..n]
2. for i=1 to n par-do
     if B[i]=true then
       for j=i+1 to n par-do
         B[j]=false
       endfor
     endif
   endfor
3. for i=1 to n par-do
     if B[i]=true then
       return i
     endif
   endfor
```

2.在PRAM-CREW模型上，用n个处理器在O(1)时间内求出数组A[1..n]={0,…,0,1,…,1}，最先为1值的下标。