Computacíon Paralela y Distribuída

2022-II

José Fiestas

29/08/22

Universidad de Ingeniería y Tecnología
jfiestas@utec.edu.pe

Unidad 2: Metodos de paralelismo

Objetivos:

1. Velocidad, eficiencia, escalabilidad. Ley de Ahmdal
2. Modelos computacionales en paralelo (PRAM)

PRAM: Suma de prefijos (algoritmo no-recursivo)

Dada una secuencia de n elementos $\{ x_1, x_2, \ldots, x_n \}$, de un conjunto S con una operación binaria asociativa (*), la suma de prefijos es:

$$s_i = x_1 * x_2 * \ldots * x_n, \quad 1 \leq i \leq n$$

Ingreso: vector A con $n = 2^k$ números

Salida : vector C tal que C(0,j) es la j-suma de prefijos para $1 \leq j \leq n$

begin

1. **for** $1 \leq j \leq n$ **pardo**

    set $B(0, j) := A(j)$

2. **for** h= 1 **to** log(n) **do**

    for $1 \leq j \leq n/2^h$ **pardo**

    set B(h,j):=B(h-1,2j-1) * B(h-1,2j)

3. **for** h= log(n) **to** 0 **do**

    for $1 \leq j \leq n/2^h$ **pardo**

    { j par : C(h,j):=C(h+1,j/2)

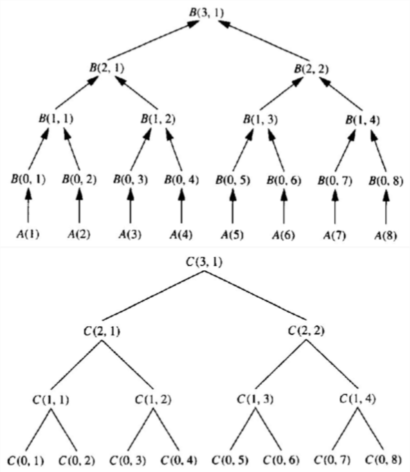
    j=1 : C(h,1):=B(h,1)

    j impar > 1 : C(h,j):=C(h+1,(j-1)/2) * B(h,j) }

end

PRAM: Suma de prefijos

Suma de prefijos de 8 elementos.



PRAM: Suma de prefijos

En el paso 1 se ejecutan n operaciones
En el paso 2 se ejecutan $\sum_{m=1}^{k} W_{2,m}$ operaciones. $W_{2,m} = n/2^m = 2^{k-m}$
, $1 \leq m \leq k$
En el paso 3 se ejecuta $\sum_{m=0}^{k} W_{3,m}$ operaciones. $W_{3,m} = 2^m$, $0 \leq m \leq k$
Por lo tanto:

$$W(n) = W_1(n) + W_2(n) + W_3(n) = n + 2^k \sum_{m=1}^{k} 2^{-m} + \sum_{m=0}^{k} 2^m$$

$$= n + n(1 - (1/n)) + 2n - 1 = O(n)$$

$$T(n) = 2O(log(n)) + 1 = O(log(n))$$

Este algoritmo ejecuta un modelo PRAM EREW

Suma de Prefijos (n,p)

PRAM para calcular la Suma de Prefijos, considerando que el número de procesos **p** es menor que la cantidad de elementos **n**. Algoritmo para el proceso p_s

Ingreso: vector A con $n = 2^k$ números, número de procesos $p = 2^q \leq n$, con índice v

Salida : vector C tal que C(0,j) es la j-suma de prefijos para $1 \leq j \leq n$

begin

1. **for** $1 \leq j \leq n/p$ **do**

   $B(0, n/p(v-1)+j) := A(n/p(v-1)+j)$

2. **for** h= 1 **to** log(n) **do**

   if (k-h-q $\geq$0) **then**

   for $n/(p*2^h)(v-1)+1 \leq j \leq n/(p*2^h)v$ **do**

   B(h,j):=B(h-1,2j-1) * B(h-1,2j)

   else if (v $\leq n/p$) **then**

   B(h,v):=B(h-1,2v-1) * B(h-1,2v)

Suma de Prefijos (n,p)

3. **for** h= log(n) **to** 0 **do**

    if (k-h-q $\geq$0) **then**

    for $n/(p*2^h)(v-1)+1 \leq j \leq n/(p*2^h)v$ **do**

        { j par   : C(h,j):=C(h+1,j/2)

        i=1   : C(h,1):=B(h,1)

        i impar $>$ 1   : C(h,j):=C(h+1,(j-1)/2) * B(h,j) }

    else if (v $\leq$ n/p) **then**

        { v par   : C(h,v):=C(h+1,v/2)

        v=1   : C(h,1):=B(h,1)

        v impar $>$ 1   : C(h,v):=C(h+1,(v-1)/2) * B(h,v) }

end

Suma de Prefijos (n,p)

Brent:

$$\frac{W(n)}{p} \leq T(n) \leq \frac{W(n)}{p} + T(n)$$

Con $W(n) = O(n)$, $T(n) = log(n)$, tenemos

$$O(n/p) \leq T(n) \leq O(n/p + log(n))$$

Entonces

$$S(n,p) = \frac{O(n)}{O(n/p + log(n))}$$

$$E(n,p) == \frac{O(n/p)}{O(n/p + log(n))} = \frac{O(1)}{O(1 + p\frac{log(n)}{n})}$$

PRAM: Suma de prefijos (algoritmo recursivo)

Dada una secuencia de n elementos $\{ x_1, x_2, \ldots, x_n \}$, de un conjunto S con una operación binaria asociativa (*), la suma de prefijos es:

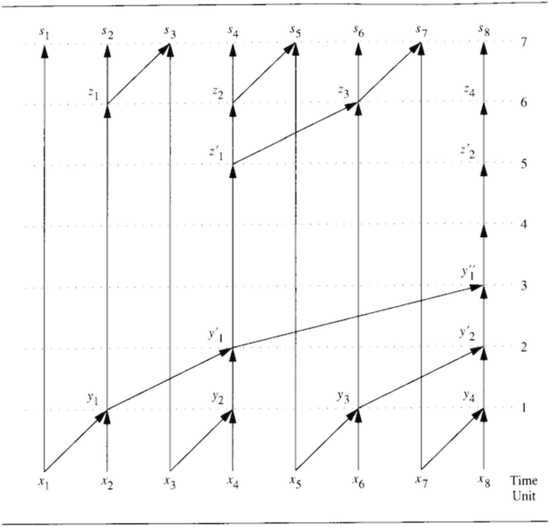
$$s_i = x_1 * x_2 * \ldots * x_n, \quad 1 \leq i \leq n$$

Ingreso: vector A con $n = 2^k$ números
Salida : suma de prefijos s_i para $1 \leq i \leq n$
begin
1. **if** n = 1 **then** $\{$set $s_1 := x_1$; **exit** $\}$
2. **for** $1 \leq i \leq n/2$ **pardo**
   set $y_i := x_{2i-1} * x_{2i}$
3. Calcular en forma recursiva, la suma de prefijos de $\{ y_1, y_2, \ldots, y_{n/2} \}$ y almacenarlos en $z_1, z_2, \ldots, z_{n/2}$
4. **for** $1 \leq i \leq n$ **pardo**
   $\{$ i par $\quad s_i = z_{i/2}$
   i=1 $\quad$: $s_1 = x_1$
   i impar $> 1 \quad s_i = z_{(i-1)/2} * x_i$ $\}$

end

PRAM: Suma de prefijos (algoritmo recursivo)

Suma de prefijos de 8 elementos.



PRAM: Suma de prefijos (algoritmo recursivo)

En el paso 1 se ejecuta 1 operación
En los pasos 2 y 4 se ejecuta 1 operación en paralelo
La recursividad tiene una complejidad dada por

$$T(n) = T(n/2) + a$$

, donde a es una constante.
La solucion de la recurrencia esta dada por: T(n)=O(log(n))

Asimismo:

$$W(n) = W_1(n) + W_2(n) + W_4(n) = O(n)$$

, dado

$$W(n) = W(n/2) + bn$$

, donde b es una constante.

Este algoritmo ejecuta un modelo PRAM EREW

PRAM: Multiplicación de matrices

Ingreso: Dos matrices A y B con nxn elementos cada una $(n = 2^l)$

Salida : El producto matricial C=A $\cdot$ B

begin

1. **for** $1 \geq i, j, k \leq n$ **pardo**

   C'(i,j,k)=A(i,k) B(k,j)

2. **for** $1 \leq h \leq log(n)$ **do**

   for $1 \geq i,\ j \leq n,\ 1 \leq k \leq n/2^h$ **pardo**

   C'(i,j,k)=C'(i,j,2k-1) + C'(i,j,2k)

3. **for** $1 \geq i, j \leq n$ **pardo**

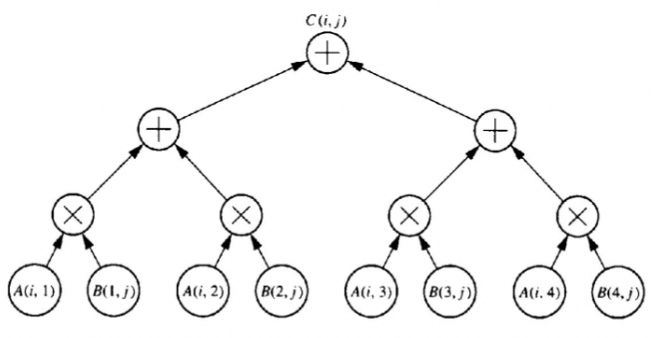
   C(i,j)=C'(i,j,1)

end

El algoritmo tiene T(n)=O(log(n)) , W(n)=O(n^3)

Aplicando el Teorema de Brent: T(n,p)=O(n^3/p+log(n))

PRAM: Multiplicación de matrices



Cálculo del elemento $C_{i,j}$ de la matriz C=AB para n=4

Bibliografía i

- Joseph Jájá *An introduction of parallel algorithms.* University of Maryland ***(Ch 2.1)
- Norm Matloff. *Programming on Parallel Machines*. University of California, Davis, 2014.
- Peter S. Pacheco. *An Introduction to Parallel Programming.* 1st. Morgan Kaufmann, 2011. isbn: 978-0-12-374260- 5.

*** en esta clase