



Programación Paralela - CC332

2021-I

José Fiestas

12/05/21

Universidad Nacional de Ingeniería
jose.fiestas@uni.edu.pe

Objetivos:

1. Velocidad, eficiencia, escalabilidad. Ley de Ahmdal
2. DAG (Directed Acyclic Graphs)
3. Modelos computacionales en paralelo (PRAM)
4. Operaciones basicas de paralelismo
5. Broadcast/Reduccion

Operaciones basicas de paralelismo

Secuencias: $a_n : \{a_0, a_1, \dots, a_{n-1}\}$

- **longitud(a)** $a = |a|$, $w=s=O(1)$
- **subsecuencia(a,i,j)**: $a = a[i, \dots, j]$, $w=s=O(1)$
- **splitmid(a)** : $(sp[0, \frac{n}{2}-1], sp[\frac{n}{2}, n-1])$, $w=O(n)$, $s=O(1)$

Tabulamiento (de una secuencia): **tab**(f(x),int,seq)

e.g. **tab**(i,n,seq), $w=O(n)$, $s=O(1)$

En general: $w = \sum_i w(f(i))$, $s = MAX(S(f(i)))$

- secuencia vacía: **tab**(f,0)

- secuencia identidad e: **tab**(f,1)

- mapping(f,a): **tab**(f(a[i]), |a|)

- append(a,b): **tab** (if $i < |a|$ then $a[i]$ else $b[i - |a|]$), $w = O(|a| + |b|)$,
 $s=O(1)$

Iteración (de una secuencia): $\text{iter}(b, f(a_i) \rightarrow x + a_i, |a|)$

E.g. $\text{iter}(0, f(a_i) \rightarrow s + a_i, |a|) \rightarrow (((((0+1)+2)+3)\dots+(n-1)) \rightarrow (((((a_0 + a_1) + a_2) + a_3)\dots + a_{n-1}))$ (suma acumulada)

E.g. la función que mapee el valor de una lista al inmediato anterior, que no sea cero.

$(1,0,4,5,2,0,0,3,4) \rightarrow (0,1,1,4,5,2,2,2,3)$

$\text{fun skipcero}(x,y) := \text{if } x > 0 \text{ then } x, \text{ else } y$

isort (a): **iter**() insert a

fun isort(x,r) = iter()

$w = \sum_i (W(f(x_i), a[i])), s = \sum_i (S(f(x_i), a[i])) = O(n^2)$

```
reduce (f,a): if  $|a| = 0$  then id  
else if  $|a| = 1$  then  $a[0]$ ,  
else (b,c)=split_mid(a)  
(rb,rc)= reduce b || reduce c  
return (rb,rc)
```

$$W(n) = 2w\left(\frac{n}{2}\right) + O(1) = O(n)$$

$$S(n) = \text{MAX}(S\left(\frac{n}{2}\right) + O(1)) = \log(n)$$

dada una función de complejidad constante $O(f(n)) = O(1)$

Broadcast / Reduction

Sea \oplus un operador asociativo, que puede ser calculado en tiempo constante, se cumple que

$$\bigoplus_{i < n} x_i := (\dots((x_0 \oplus x_1) \oplus x_2) \oplus \dots \oplus x_{n-1})$$

Se calcula en tiempo $O(\log n)$ en PRAM, y en $O(T_{start} \log n)$ en un array lineal.

E.g., $+$, \cdot , \max , \min

Operaciones asociativas, reducción

código PRAM ($p=n$)

$p \in \{0, \dots, n-1\}$

activo:=1

for $0 \leq k < \log n$ **do**

if activo **then**

if bit k of i **then**

 activo=0

else if $i + 2^k < n$ **then**

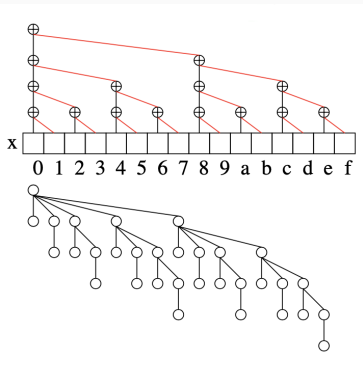
$x_i := x_i \oplus x_{i+2^k}$

los resultados aparecen en x_0

Dados n

procesos, $T=O(\log n)$, velocidad

$S=O(n/\log n)$, $E=O(1/\log n)$



Operaciones asociativas, reducción

$$p < n$$

n/p elementos a cada proceso

Entonces, la suma es

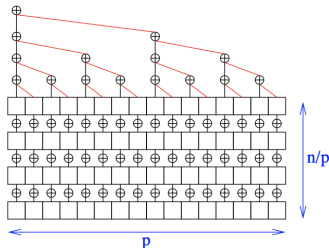
en paralelo de p sumas parciales

Tiempo = $T_{seq} (n/p) + O(\log p)$

Eficiencia:

$$\frac{T_{seq}}{p(T_{seq}(n/p) + O(\log p))} = \frac{1}{1 + O(p \log p)/n} = 1 - O\left(\frac{p \log p}{n}\right),$$

si $n \gg p \log p$



Operaciones asociativas, reducción

Memoria distribuida

$p \in \{0, \dots, n-1\}$

activo:=1

$s : x_i$

for $0 \leq k \leq \log n$ **do**

if activo **then**

if bit k of i **then**

sync-send s to $p \ i - 2^k$

 activo=0

else if $i + 2^k < n$ **then**

receive s' from $p \ i + 2^k$






$s := s \oplus s'$

los resultados aparecen en $p=0$

Comunicación total: $\Theta((T_{start} + T_{byte}) \log p)$

Array lineal: $\Theta(p)$, paso k necesita 2^k

BSP: $\Theta((l+g) \log p) = \Omega(\log^2 p)$

-  David B. Kirk and Wen-mei W. Hwu *Programming Massively Parallel Processors: A Hands-on Approach*. 2nd. Morgan Kaufmann, 2013. isbn: 978-0-12-415992-1.
-  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.
-  Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. 1st. McGraw-Hill Education Group, 2003. isbn: 0071232656.
-  Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Program- ming*. 1st. Addison-Wesley Professional, 2010. isbn: 0131387685, 9780131387683.