Computacíon Paralela y Distribuída

2022-II

José Fiestas

31/08/22

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

Unidad 2: Metodos de paralelismo

Objetivos:

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

Operaciones basicas de paralelismo

Operaciones en secuencias

Procesamiento de secuencias es una de las principales aplicaciones de algoritmos paralelos

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

- **indexado(a)** $a[0] = a_0$, $a[1] = a_1$, ... W=T=O(1)
- **longitud(a)** $a = |a|$, W=T=O(1)
- **subsec(a,i,j)**: $a = a[i, ..., j]$, W=T=O(1)
- **splitmid(a)** : (subsec[0,$\frac{n}{2}$-1],subsec[$\frac{n}{2}$,n-1]), W=O(n), T=O(1)

Tabular (map)

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

e.g. **tab**(i,seq), W=O(n), T=O(1)
En general: $W = \sum_i W(f(i))$, $T = MAX(T(f(i)))$
- 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($|a| + |b|$), T=O(1)

map (left, right, f, in, out):

input: array *in*, funcion *f*, posiciones extremas *left*, *right*
output: array *out*

```
i:= left
while (i < right){
    out(i)=f(inp(i))
    i=i+1
}
```

T(n)=O(n)
W(n)=O(n)

map (left, right, f, in, out):

input: array *in*, funcion *f*, posiciones extremas *left*, *right*
output: array *out*

```
 if (right - left < threshold)
    map (left, right, f, in, out)
else
    mid= left + (right -left)/2
    map(left, mid, f, in, out) || map (mid, right, f, in, out)
```

T(n)=MAX(T($\frac{n}{2}$)+O(1))=log(n)
W(n)=2w($\frac{n}{2}$)+O(1)=O(n)
dada una función de complejidad constante O(f(n))=O(1)

Recurrencias

Dado f(1)=0; f(n)=f(n-1)+n (reescribir f(n-1) como f(n-1)-1)+n-1)
= f(n-2)+n-1+n
= f(n-3)+n-2+n-1+n
. . .
= f(1)+f(2)+f(3)+ . . . + n-1+n
= n$\frac{(n+1)}{2}$-1 = $\Theta(n^2)$

---

f(n)=f(n/2)+1 (reescribir f(n/2) como f((n/2)/2)+1)
= f(n/4)+1+1
= f(n/8)+1+1+1
. . .
= f(n/n)+1+1+ . . . + 1 (log(n) veces)
= $\Theta(log(n))$

Recurrencias

Dado f(1)=0; f(n)=f(n/2)+n
$= f(n/4)+n/2+n$
$= f(n/8)+n/4+n/2+n$
. . .
$= f(n/n)+2+4+ \ldots + n/4+n/2+n$
$= \Theta(n)$

---

f(n)=2f(n/2)+n
$= 4f(n/4)+n+n$
$= 8f(n/8)+n+n+n$
. . .
$= nf(n/n)+n+n+ \ldots +n$ (log(n) veces)
$= \Theta(nlog(n))$

exp_array (left, right, f, in, out):

input: array *in*, funcion *f*, posiciones extremas *left*, *right*

output: array *out*

```
i:= left
while (i < right){
    out(i)=power(inp(i))
    i=i+1
}
```

T(n)=O(n)

W(n)=O(n)

exp_array_par (left, right, f, in, out):

input: array *in*, funcion *f*, posiciones extremas *left*, *right*

output: array *out*

```
 if (right - left < threshold)
    exp_array (left, right, f, in, out)
else
    mid= left + (right -left)/2
    exp_array(left, mid, f, in, out) || exp_array (mid, right, f,
```

T(n)=MAX(T($\frac{n}{2}$)+O(1))=log(n)

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

Iteración (fold)

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

E.g. **iter**(0,$f(a_i) \rightarrow s + a_i, |a|$) $\rightarrow$ ((((0+1)+2)+3)...+(n-1)) $\rightarrow$
$((((a_0 + a_1) + a_2) + a_3)... + a_{n-1})$ (suma acumulada)
E.g. **iter**(0,$f(a_i) \rightarrow$ *left*(100)($s - a_i, |a|$) $\rightarrow$
((((100-a_1)-a_2)-a_3)...-$a_{(n-1)}$)
E.g. **iter**(0,$f(a_i) \rightarrow$ *right*(100)($s - a_i, |a|$) $\rightarrow$
(a_1-(a_2-(a_3-...-($a_{(n-1)}$-100))))

reduce (left, right, f, in, res):

input: array *in*, funcion *f*, posiciones extremas *left*, *right*
output: *res*

```
res:= res0
i:= left
while (i < right){
    res:=f(res, inp(i))
    i=i+1
}
```

T(n)=O(n)
W(n)=O(n)

reduce (left, right, f, in, res):

input: array *in*, funcion *f*, posiciones extremas *left*, *right*

output: *res*

```
 if (right - left < threshold)
    reduce (left, right, f, in, res)
else
    mid= left + (right -left)/2
    reduce(left, mid, f, in, res) || reduce(mid, right, f, in, res
```

$\mathrm{T(n)}=\mathrm{MAX}(\mathrm{T}(\frac{n}{2})+\mathrm{O}(1))=\log(\mathrm{n})$

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

scan

$\boxed{\textbf{scan} \text{ (de una secuencia): } \textbf{iter}(a, \textbf{map}(f(a_i) \rightarrow x + a_i), |a|\ )}$

E.g. scanleft(a,(100)(s+x)) $\rightarrow$ (100, 100+a_1, 100+a_1+a_2, . . . , 100+a_1+ . . . + $a_{(n-1)}$)

scan_left (left, right, f, in, out):

: array *in*, funcion *f*, posiciones extremas *left*, *right*

output: array *out*

```
out(0)=a0
a:=a0
i:=0
while (i < inp.size()){
    a:=f(a,inp(i))
    i=i+1
    cout(i):=a
}
```

T(n)=O(n)
W(n)=O(n)

scan_left (f, in, out):

input: array *in*, funcion *fi*, posiciones extremas *0*, *in.size()*
output: array *out*

```
fi:=reduce(0, i, f, in, f)
map (0, inp.size(), fi, in, out)
last=inp.size()-1
out(last+1)=f(out(last),inp(last))
```

$W(n)=2w(\frac{n}{2})+O(1)=O(n)$
$T(n)=MAX(T(\frac{n}{2})+O(1))=\log(n)$

Bibliografía i

- 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.