Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 11 - "Algoritmos Paralelos"

Profesor: Pablo Barceló
Auxiliar: ~~Manuel~~ Ariel Cáceres Reyes

16 de Junio del 2017

P1. Copias

Se desea, dado un cierto valor x, generar n copias de este.

a) Diseñe un algoritmo CREW de tiempo $\mathcal{O}(1)$ y eficiencia $\Theta(1)$

b) Diseñe un algoritmo EREW de tiempo $\mathcal{O}(\log n)$ que use n procesadores

c) Mejórelo para obtener eficiencia $\Theta(1)$

P2. Multiplicar Matrices

Considere el problema de multiplicar dos matrices de $n \times n$. Considere que $T(n) = \mathcal{O}(n^3)$, es decir, ignoramos algoritmos mejores al estándar.

a) Proponga un algoritmo de tiempo $\mathcal{O}(n)$ usando n^2 procesadores. Calcule $T(n,p), W(n), E(n,p)$. ¿Qué modelo PRAM usa?

b) Proponga un algoritmo de tiempo $\mathcal{O}(\log n)$ usando n^3 procesadores. Calcule $T(n,p), W(n), E(n,p)$

c) Mejore la eficiencia del algoritmo anterior para que sea $\Theta(1)$. ¿Qué tiempo obtiene y cuántos procesadores utiliza?

P3. Select(A, 1)

Sea A un arreglo de bits de largo n. Diseñe un algoritmo CRCW que tome tiempo $\mathcal{O}(1)$ utilizando $\mathcal{O}(n)$ procesadores para encontrar Select(A, 1), el primer elemento k tal que $A[k] = 1$.

P1. a) Los n procesadores miran a x y lo copian en $B[p]$ en 1 paso paralelo. Con esto tenemos que:

$$\begin{aligned} T(n,n) &= 1 \\ T(n) &= n \\ E(n,n) &= \frac{n}{n \cdot 1} = \mathcal{O}(1) \end{aligned}$$

b) Lo anterior es CREW, para hacerlo EREW duplicamos los x's en cada paso usando el doble de procesadores que en el paso anterior, hasta $\frac{n}{2}$.

```
for i ← 0 ... (log(n) − 1) do
    if p ≤ 2^i then
        B[2^i + p] = B[p]
    end
end
```

Notemos ahora que son $\log n$ pasos y se ocupan $\frac{n}{2}$ procesadores, por lo que:

$$\begin{aligned} T(n, n/2) &= \log n \\ E(n, n/2) &= \frac{n}{n/2 \cdot \log n} = \mathcal{O}(1/\log n) \end{aligned}$$

c) ¿Primero veamos que $W(n) = n$ pues se siguen copiando la misma cantidad de elementos. Ahora, por lema de Brent tenemos que existe un algotimos EREW tal que:

$$\begin{aligned} T\left(n, \frac{n}{\log n}\right) &= \mathcal{O}(\log n) \\ E\left(n, \frac{n}{\log n}\right) &= \frac{n}{\frac{n}{\log n} \cdot \mathcal{O}(\log n)} = \mathcal{O}(1) \end{aligned}$$

P2. a) Si tenemos n^2 procesadores a cada uno le asignamos el cálculo de $C_{i,j}$ y por lo tanto, podemos hacer la multiplicación en $\mathcal{O}(n)$ pasos paralelos (lo que tarda un producto punto). Luego:

$$\begin{aligned} T(n,n) &= \mathcal{O}(n) \\ E(n,n) &= \frac{n^3}{n^2 \cdot \mathcal{O}(n)} = \mathcal{O}(1) \end{aligned}$$

Notar que este es un algoritmo CREW pues hay procesadores que miran a la misma fila y columna.

b) Con n^3 procesadores:

- En un paso paralelo calculamos $A_{i,k} \cdot B_{k,j}, \forall i,j,k$ y lo ponemos en $C_{i,k,j}$.
- Queremos calcular $C_{i,j} = \sum_{k \in [n]} C_{i,k,j}$ en $\mathcal{O}(\log n)$ pasos.

Para realizar esto asignaremos n procesadores a cada par (i,j) quienes se encargarán de hacervla suma de estos n elementos. Para alcanzar el $\mathcal{O}(\log n)$ aplicaremos la misma idea de "torneo" utilizada para el problema de encontrar el máximo en un arreglo, según el siguiente código:

```
for d ← 0 ... (log(n) − 1) do
    if (p − 1)%2^{d+1} = 0 then
        C[i][p][j] = C[i][p][j] + C[i][p + 2^d][j]
    end
end
```

Con esto tenemos:

$$T(n, n^3) = \log n$$
$$E(n, n^3) = \frac{n^3}{n^3 \cdot \log n} = \mathcal{O}(1/\log n)$$

c) Veamos que $W(n) = n^3$, pues las multiplicaciones son n^3 y después hacemos n^2 sumas, cada una considerando n sumandos.
Lamentablemente, aunque las sumas son EREW, el cálculo de los $C_{i,k,j}$ no lo es. Para poder aplicar el lema de Brent solucionaremos este último problema.
Para el cálculo de un $C_{i,k,j} = A_{i,k} \cdot B_{k,j}$, tendremos n procesadores revisando una misma casilla de A y n revisando una misma casilla de B, por lo tanto haciendo n copias de cada una de las matrices tenemos el problema solucionado.
Como vimos en el P1, podemos hacer n copias de un elemento en $\log n$ pasos paralelos con n procesadores, de este modo, podemos copiar una A en $\log n$ pasos paralelos con n^3 procesadores, lo que no aumenta la complejidad (paralela) del problema

P3. Dividimos el bitmap en bloques de tamaño $\sqrt{n}$, creando un bitmap

$$B[i] = \begin{cases} 0 & si\ A[(i-1)\sqrt{n}, i\sqrt{n}] = 0s \\ 1 & si\ \exists 1 \in A[(i-1)\sqrt{n}, i\sqrt{n}] \end{cases}$$

- B puede ser llenado en 1 paso paralelo (los n procesadores miran cada uno una casilla de A y si ven un 1 ponen un 1 en la casilla correspondiente de B.
- En otro paso paralelo podemos identificar el primer bloque que contiene un 1, sea i^* este bloque.
- Finalmente, obtenemos el primer uno del bloque i^*, llamándole j^* con lo que el resultado sería $(i^* - 1)\sqrt{n} + j^*$.

Para hacer esto último aplicamos un algoritmo bárbaro, que tiene un procesador mirando cada par de elementos del bloque (como el bloque es de tamaño $\sqrt{n}$ existen $\mathcal{O}(\sqrt{n}^2) = \mathcal{O}(n)$ pares) y si ambos elementos del par tienen un 1 escribe un 0 en el elemento de más a la derecha, de este modo, al finalizar, el único que tendrá un 1 en el bloque será j^*.