Algoritmos **paralelos** y distribuidos (2w = 4t = 360mns)

Jérémy Barbay

21 de junio de 2012

- Medidas de complejidad
- Tecnicas de diseno

1. PREREQUISITOS

- Chap 12 of "Introduction to Algorithms, A Creative Approach", Udi Manber, p. 375
- http://www.catonmat.net/blog/mit-introduction-to-algorithms-part-thirteen

2. Modelos de paralelismo y modelo PRAM

- Instrucciones
  - SIMD: Single Instruccion, Multiple Data
  - MIMD: Multiple Instruccion, Multiple Data
- Memoria
  - compartida
  - distribuida
- 2*2 combinaciones posibles:

	Memoria compartida	Memoria distribuida
SIMD	PRAM	redes de interconexion (weak computer units) (hipercubos, meshes, etc...)
MIMD	Threads	procesamiento distribuido (strong computer units), Bulk Synchronous Process, etc...

2.1. Modelo PRAM

En este curso consideramos en particular el modelo PRAM.

- Mucha unidad de CPU, una sola memoria RAM
  cada procesador tiene un identificador unico, y puede utilizarlo en el programa
- Ejemplo:
  - if $p2 = 0$ then
    - $A[p] += A[p-1]$
  - else

    - $A[p] += A[p+1]$
  - $b \leftarrow A[p]$;
  - $A[p] \leftarrow b$;

- Problema: el resultado no es bien definido, puede tener **conflictos** si los procesadores estan asinchronos. Las soluciones a este problemas dan varios submodelos del modelo PRAM:

  1. EREW Exclusive Read. Exclusive Write
  2. CREW Concurrent Read, Exclusive Write
  3. CRCW Concurrent Read, Concurent Write En este caso hay variantes tambien:
     - todos deben escribir lo mismo
     - arbitrario resultado
     - priorizado
     - alguna $f()$ de lo que se escribe

2.2. Como medir el "trade-off" entre recursos (cantidad de procesadores) y tiempo?

- DEFINICION:

  - $T^*(n)$ es el **Tiempo secuencial** del mejor algoritmo no paralelo en una entrada de tamano n (i.e. usando 1 procesador).
  - $T_A(n,p)$ es el **Tiempo paralelo** del algoritmo paralelo A en una entrada de tamano n usando p procesadores.
  - El **Speedup** del algoritmo A es definido por
    $S_A(n,p) = \frac{T^*(n)}{T_A(n,p)} \leq p$
    Un algoritmo es mas efectivo cuando $S(p) = p$, que se llama **speedup perfecto**.
  - La **Eficiencia** del algoritmo A es definida por
    $E_A(n,p) = \frac{S_A(n,p)}{p} = \frac{T^*(n)}{pT_A(n,p)}$
    El caso optima es cuando $E_A(n,p) = 1$, cuando el algoritmo paralelo hace la misma cantidad de trabajo que el algoritmo secuencial. El objetivo es de **maximizar la eficiencia**.
    (Nota estas definiciones en la pisara, vamos a usarlas despues.)

3. LEMMA de Brent, Trabajo y Consecuencias

3.1. PROBLEMA: Calcular $\max(A[1,...,N])$

1. Solucion Secuencial

   - Algoritmo:
     - $m \leftarrow 1$
     - for $i \leftarrow 2$ to n
       - if $A[i] > A[m]$ then $m \leftarrow i$
     - return $A[m]$
   - Se puede ver como un arbol de evaluacion con una sola rama de largo n y n hojas.
   - Complejidad:
     - tiempo $O(n)$, con 1 procesador, entonces:
     - $T^*(n) = n$.

2. Solucion Parallela con n procesadores
   - Algoritmo:
     - $M[p] \leftarrow A[p]$
     - for $l \leftarrow 0$ to $\lceil \lg p \rceil - 1$
       - if $p2^{l+1} = 0$ y $ p+2^l<n$
         - $M[p] \leftarrow \max(M[p], M[p + 2^l])$
     - if $p = 0$
       - $max \leftarrow M[0]$
   - Se puede ver como un arbol balanceado de altura $\lg n$ con n hojas.
   - Complejidad:
     - tiempo $O(\lg n)$ con n procesador, i.e. en nuestras notaciones:
     - $T(n, n) = \lg n$
     - $S(n, n) = \frac{n}{\lg n}$
     - $E(n, n) = \frac{n}{n \lg n} = \frac{1}{\lg n}$
   - Nota: no se puede hacer mas rapido, pero hay mucho procesadores poco usados: quizas se puede calcular el max en el mismo tiempo, pero usando menos procesadores?
3. Solucion general con p procesadores
   - Idea:
     - reduce la cantidad de procesadores, y hace "load balancing" sobre $n/\lg n$ procesadores.
     - Divida el input en $n/\lg n$ grupos,
     - asigna cada grupo de $\lg n$ elementos a un procesador.
     - En la primera fase, cada procesador encontra el max de su grupo
     - En la segunda fase, utiliza el algoritmo precedente.
   - Complejidad:
     - tiempo $O(\lg n)$ con n procesador, i.e. en nuestra notaciones:
       - $T(n, \frac{n}{\lg n}) = 2 \lg n \in O(\lg n)$
         - $T(n, p) = \frac{n}{p} + \lg p$
       - $S(n, p) = \frac{n}{\frac{n}{p} + \lg p} = p(1 - \frac{p \lg p}{n + p \lg p}) \rightarrow p$ si $n \rightarrow \infty$
       - $E(n, p) = \frac{n}{\frac{n}{\lg n} \lg n} = 1/2$
   - Discusion:
     - El parametro de $\lg n$ procesadores es optima?
       - Para que? Que significa ser optima?
         - en energia
         - en el contexto donde los procesadores libres pueden ser usados para otras tareas.
       - Si, es optimo para la eficiencia, se puede ver estudiando el grafo en funcion de p.
     - Eso es un algoritmo EREW, CREW, o CRCW?
       - EREW (Exclusive Read. Exclusive Write): no dos procesadores lean o escriben en la misma cedula al mismo tiempo.
     - Nota:
       - Hay un algoritmo CRCW que puede calcular el max en $O(1)$ tiempo en paralelo, ilustrando el poder del modelo CRCW (y el costo de las restricciones del modelo EREW) [REFERENCIA: Section 12.3.2 of "Introduction to Algorithms, A Creative Approach", Udi Manber, p. 382]]

3.2. LEMA de Brent

El algoritmo previo illustra un principo mas general, llamado el "Lemma de Brent":
Si un algoritmo

- consigue un tiempo T(n,p)=C, entonces
- consigue tiempo T(n,p/s)= sC $\forall$ s>1
- (bajo algunas condiciones, tal que hay suficientamente memoria para cada procesador)

3.3. DEFINICION "Trabajo"

- Usando el Lema de Brent, podemos exprimir el rendimiento de los algoritmos paralelos con solamente dos medidas:
  - $T(n)$, el tiempo del mejor algoritmo paralelo usando cualquier cantidad de procesadores que quiere. Nota las diferencias con
    - $T^*(n)$, el tiempo del mejor algoritmo secuencial, y
    - $T_A(n, n)$, el tiempo del algoritmo A con n procesadores.
  - $W(n)$, la suma del total trabajo ejecutado por todo los procesadores (i.e. superficia del arbol de calculo, a contras de su altura (tiempo) o hancho (cantidad de procesadores).
- INTERACCION: Cual son estas valores para el algoritmo de Max?
  - $T(n)$ =?
  - $W(n)$ =?
- INTERACCION: Puedes ver como desde $T(n)$, $W(n)$ se puede deducir las valores de
  - $T(n, p)$? (solucion en el corolario)
  - $S(n, p)$? (trivial desde $T(n, p)$)
  - $E(n, p)$? (solucion en el corolario)

3.4. COROLARIO

- Con el lema de Brent podemos obtener:
  - $$T(n,p) = T(n) + \frac{W(n)}{p}$$
  - $$E(n,p) = \frac{T^*(n)}{pT(n) + W(n)}$$

3.5. EJEMPLO

- Para el calculo del maximo:
  - $T(n) = \lg n$
  - $W(n) = n$
- Entonces
  - se puede obtener
    - $T_B(n, p) = \lg n + \frac{n}{p}$
    - $$E(n,p) = \frac{n}{p \lg n + n}$$
- (Nota que eso es solamente una cota superior, nuestro algoritmo da un mejor tiempo.)

4. PROBLEMA: Ranking en listas

1. DEFINICION
   - dado una lista, calcula el rango para cada elemento.
   - En el caso de una lista tradicional, no se puede hacer mucho mejor que lineal.
   - Consideramos una lista en un arreglo A,
     - donde cada elemento $A[i]$ tiene un puntero al siguiente elemento, $N[i]$, y
     - calculamos su rango $R[i]$ en un arreglo R.
2. DoublingRank()
   - $R[p] \leftarrow 0$
   - if $N[p] = null$
     - $R[p] \leftarrow 1$
   - for $d \leftarrow 1$ to $\lceil \lg n \rceil$
     - if $N[p] \neq NULL$
       - if $R[N[p]] > 0$
         - $R[p] \leftarrow R[N[p]] + 2^d$
       - $N[p] \leftarrow N[N[p]]$
3. Analisis
   - $T(n) = \lg n$
   - $W(n) = n + W(n/2) \in O(n)$
   - $T(n,p) = T(n) + W(n)/p = \lg n + n/p$
   - $p^*\ T(n) = W(n)/p^*\ pn/\lg n$
   - $E(n,p^*) = \frac{T^*(n)}{p^*T(n)+W(n)} = \frac{n}{n/\lg n \lg n+n} \in \Theta(1)$
4. El algoritmo es EREW o CREW?
   - es EREW si los procesadores estan sincronizados, com en RAM aqua.

5. PROBLEMA: Prefijos en paralelo ("Parallel Prefix")

- DEFINICION: Problema "Prefijo en Paralelo"

  Dado $x_1, \ldots, x_n$ y un operador asociativo $\times$, calcular
  - $y_1 = x_1$
  - $y_2 = x_1 \times x_2$
  - $y_2 = x_1 \times x_2 \times x_3$
  - $\ldots$
  - $y_n = x_1 \times \ldots \times x_n$
- Solucion Secuencial

  Hay una solucion obvio en tiempo $O(n)$.

5.1. Solucion paralela 1

- Concepto:
  - Hipotesis: sabemos solucionarlo con $n/2$ elementos
  - Caso de base: $n = 1$ es simple.
  - Induccion:
    1. recursivamente calculamos en paralelo:
       - todos los prefijos de $\{x_1, \ldots, x_{n/2}\}$ con $n/2$ procesadores.
       - todos los prefijos de $\{x_{n/2}, \ldots, x_n\}$ con $n/2$ procesadores.
    2. en paralelo agregamos $x_{n/2}$ a los prefijos de $\{x_{n/2}, \ldots, x_n\}$
- Observacion: en cual modelo de parallelismo es el ultimo paso?
- ParallelPrefix1(i,j)
  - if $i_p = j_p$
    - return x_{i_p}
  - $m_p \leftarrow \lfloor \frac{i_p + j_p}{2} \rfloor$;
  - if $p \leq m$ then
    - algo(i_p, m_p)
  - else
    - algo$(m + 1, j_p)$
    - $y_p \leftarrow y_m . y_p$
- Otra forma de escribir el algoritmo (de p. 384 de "Introduction to Algorithms, A Creative Approach", Udi Manber):
  - ParallelPrefix1(left,right)
    - if $(right - left) = 1$
      - $x[right] \leftarrow x[left].x[right]$
    - else
      - $middle \leftarrow (left + right - 1)/2$
      - do in paralel
      - $ParallelPrefix1(left, middle)\{$ assigned to $\{P_1 to P_{n/2}\}\}$
      - $ParallelPrefix1(middle + 1, right)\{$ assigned to $\{P_{n/2+1} to P_n\}\}$
      - for $i \leftarrow middle + 1$ to $right$ do in paralel
      - $x[i] \leftarrow x[middle].x[i]$
- Notas:
  - este solucion **no** es EREW (Exclusive Read and Write), porque los procesadores pueden leer y_m al mismo tiempo.
  - este soluciono es CREW (Concurrent Read, Exclusive Write).
  - Complejidad:
    - $T_{A_1}(n, n) = 1 + T(n/2, n/2) = \lg n$
      (El mejor tiempo en paralelo con cualquier cantidad de procesadores.)
    - $W_{A_1}(n) = n + 2W_{A_1}(n/2) - n \lg n$

- $T_{A_1}(n,p) = T(n) + W_{A_1}(n)/p = \lg n + (n \lg n)/p$
- Calculamos p^*, la cantidad optima de procesadores para minimizar el tiempo:
  - $T(n) = W_{A_1}(n)/p^*$
  - $p^* = \frac{W(n)}{T(n)} = n$
- Calculamos la eficiencia
  - $E_{A_1}(n,p^*) = \frac{T^*(n)}{p^* T(n) + W(n)} = \frac{n}{n \lg n} = \frac{1}{\lg n}$
  - es poco eficiente =(
  - Podriamos tener un algoritmo con
  - la eficiencia del algoritmo secuencial
  - el tiempo del algoritmo paralelo?

5.2. Solucion paralela 2: mismo tiempo, mejor eficiencia

- Idea:

  El concepto es de dividir de manera diferente: par y impar (en vez de largo o pequeno).

- Concepto:

  1. Calcular en paralelo $x_{2i-1}.x_{2i}$ en x_{2i} para $\forall i, 1 \leq i \leq n/2$.
  2. Recursivamente, calcular todos los prefijos de $E = \{x_2, x_4, \ldots, x_{2i}, \ldots, x_n\}$
  3. Calcular en paralelo los prefijos en posiciones impares, multiplicando los prefijos de E por una sola valor.

- algo2(i,j)

  - for $d \leftarrow 1$ to $(\lg n) - 1$
    - if $p = 02^{d+1}$
      - if $p + 2^d < n$
      - $x_{p+2^d} \leftarrow x_p.x_{p+2^d}$
  - for $d \leftarrow 1$ to $(\lg n) - 1$
    - if $p = 02^{d+1}$
      - if $p - 2^d > 0$
      - $x_{p-2^d} \leftarrow x_{p-2^d}.x_p$

- visualizacion

  1.
  2. (0,1)
  3.
  4. (2,3) (0,3)

5.
6. (4,5)
7.
8. (6,7) (4,7) (0,7)
9.
10. (8,9)
11.
12. (10,11) (8,11)
13.
14. (12,13)
15.
16. (14,15) (12,15) (8,15) (0,15)

- Notas:
  - Este algoritmo es EREW
- Análisis
  - $T_2(n) = 2 \lg n$
  - $W(n) = n/2 + W(n/2) < n$
  - $T_2(n,p) = T_2(n) + W(n)/p < 2 \lg n + \frac{n}{p}$
  - Calculamos la cantidad optima de procesadores para obtener el tiempo optima:
    - $T_2(n) = W(n)/p^*$ entonces
    - $p^* = \frac{W(n)}{T(n)} = \frac{n}{\lg n}$
  - $E_2(n,p^*) = \frac{T^*(n)}{p^* T(n) + W(n)} \in O(1)$

6. Moralidad del Parallelismo:

1. Cual es la consecuencia del Lemma de Brent?
   - Concentrarse en $T(n)$ y $E(n)$
   - Pero saber aplicar el Lemma de Brent para programmar $T(n,p)$
2. Cual (otra) tecnica veamos?
   - Mejorar la efficiencia con una parte secuencial (en parallelo) del algoritmo.