Multi-threaded Algorithm 3

Michael Tsai

2012/1/3

Multithreaded matrix multiplication

```
P-SQUARE-MATRIX-MULTIPLY(A,B)
n=A.rows
let C be a new n x n matrix
parallel for i=1 to n
    parallel for j=1 to n
        c_ij = 0
        for k=1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

$T_1(n) = \Theta(n^3)$

$T_\infty(n) = \Theta(\log n) + \Theta(\log n) + \Theta(n) = \Theta(n)$

$$\frac{T_1(n)}{T_\infty(n)} = \frac{\Theta(n^3)}{\Theta(n)} = \Theta(n^2)$$

Divide-and-conquer Multithreaded Algorithm for Matrix Multiplication (勒勒長)

- $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$
- $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$
- $C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$

$$= \begin{pmatrix} A_{11}B_{11} & A_{12}B_{12} \\ A_{21}B_{11} & A_{22}B_{12} \end{pmatrix} + \begin{pmatrix} A_{12}B_{21} & A_{12}B_{22} \\ A_{22}B_{21} & A_{22}B_{22} \end{pmatrix}$$

C T

```
P-MATRIX-MULTIPLY-RECURSIVE(C,A,B)
n=A.rows
if n==1
        c_11 = a_11 b_11
else let T be a new n x n matrix
        partition A,B,C, and T into n/2 x n/2
submatrices (A_ij, B_ij, C_ij, T_ij, i,j = {1,2})
        spawn P-MATRIX-MULTIPLY-RECURSIVE(C_11, A_11, B_11)
        spawn P-MATRIX-MULTIPLY-RECURSIVE(C_12, A_11, B_12)
        spawn P-MATRIX-MULTIPLY-RECURSIVE(C_21, A_21, B_11)
        spawn P-MATRIX-MULTIPLY-RECURSIVE(C_22, A_21, B_12)
        spawn P-MATRIX-MULTIPLY-RECURSIVE(T_11, A_12, B_21)
        spawn P-MATRIX-MULTIPLY-RECURSIVE(T_12, A_12, B_22)
        spawn P-MATRIX-MULTIPLY-RECURSIVE(T_21, A_22, B_21)
        P-MATRIX-MULTIPLY-RECURSIVE(T_22, A_22, B_22)
        sync
        parallel for i=1 to n
                parallel for j=1 to n
                        c_ij = c_ij + t_ij
```

$$M_1(n) = 8M_1\left(\frac{n}{2}\right) + \Theta(n^2) = \Theta(n^3)$$

$$\frac{M_1(n)}{M_\infty(n)} = \frac{\Theta(n^3)}{\Theta(\log^2 n)} = \Theta\left(\frac{n^3}{\log^2 n}\right)$$

$$M_\infty(n) = M_\infty\left(\frac{n}{2}\right) + \Theta(\log n) + \Theta(\log n) = \Theta(\log^2 n)$$

How about Strassen's method?

- Reading assignment: p.795-796.

- Parallelism: $\Theta(\frac{n^{\log 7}}{\log^2 n})$, slightly less than the original recursive version!

Multithreaded Merge Sort

```
Merge-Sort'(A,p,r)
if p<r
    q=⌊(p+r)/2⌋
    spawn MERGE-SORT'(A,p,q)
    MERGE-SORT'(A,q+1,r)
    sync
    MERGE(A,p,q,r)
```

A: Array to be sorted
p and r: Start and end index of the range to be sorted

Merge() is the bottleneck!

$\Theta(n)$

$$MS'_1(n) = 2MS'_1\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n \log n)$$

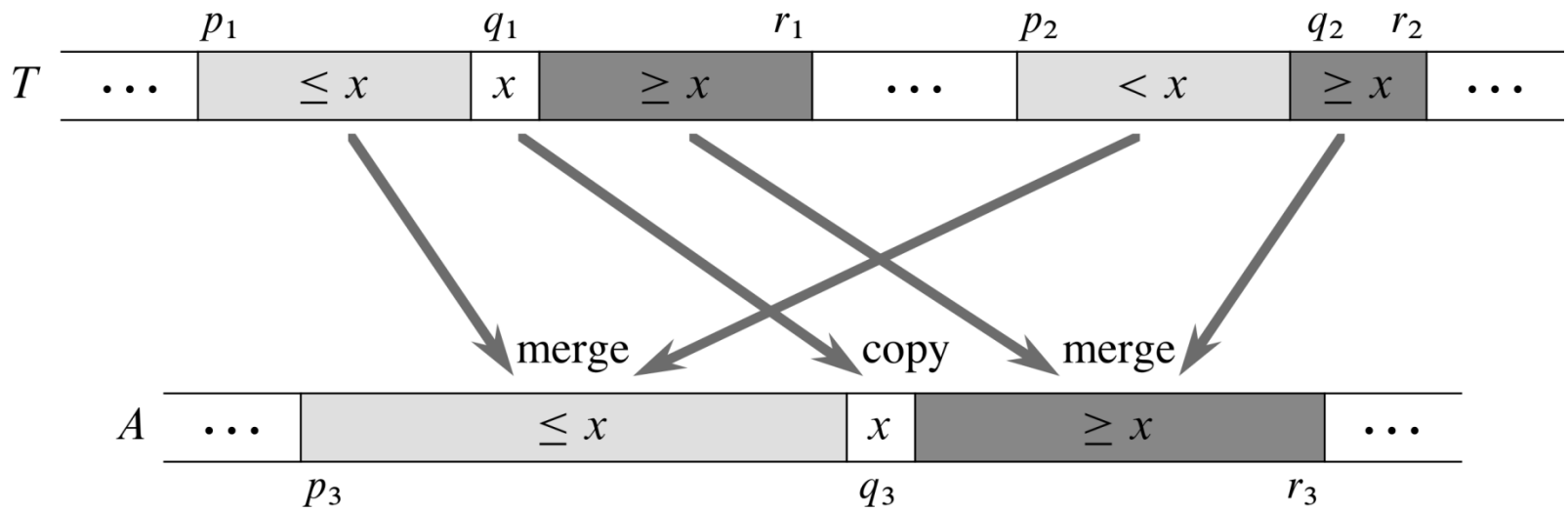
Parallelism= $\Theta(\log n)$

$$MS'_\infty(n) = MS'_\infty\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n)$$

把Merge也用Divide & Conquer來解!
(方便交給不同的thread去做)

1. 挑出$p_1 \sim r_1$的中位數x

2. 找出$p_2 \sim r_2$中q_2這個位置使得$p_2 \sim q_2 - 1$都< x, $q_2 \sim r_2$都≥ x



3. Copy x 到新地方(根據x和q_2的位置)

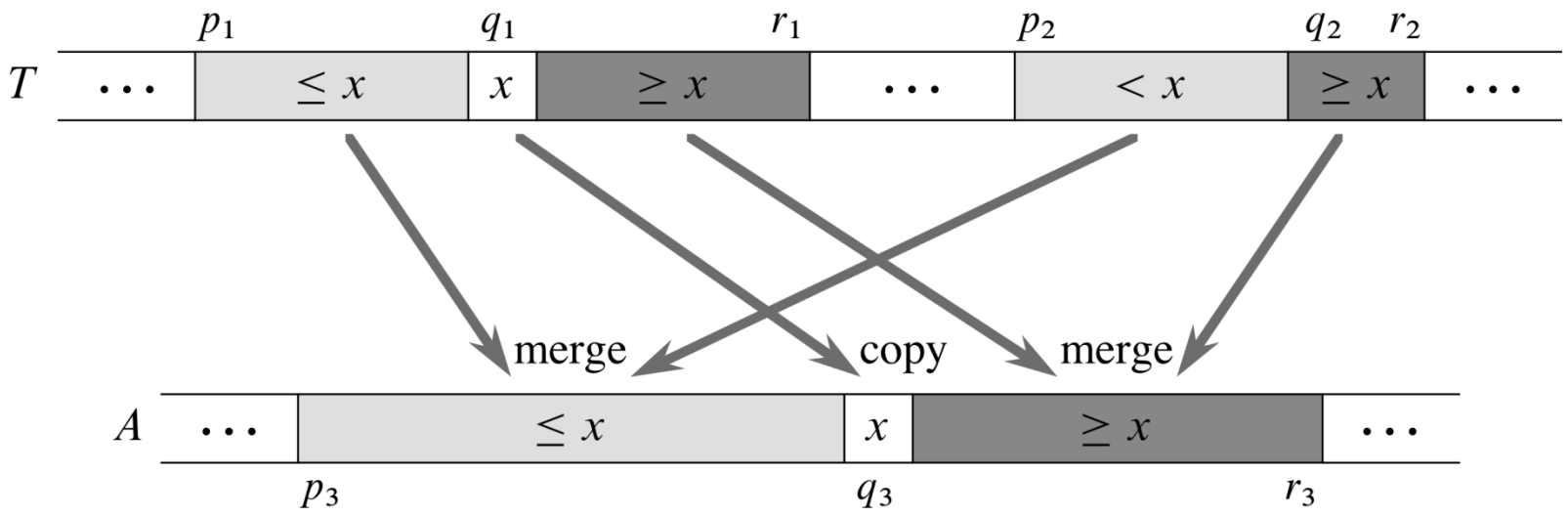
4. 開分身去merge
$p_1 \sim q_1 - 1$和$p_2 \sim q_2 - 1$, 及
$q_1 + 1 \sim r_1$和$q_2 \sim r_2$兩個區段

Base case:
$p_1 \sim r_1$和$p_2 \sim r_2$都是空的

Multithreaded Merge

```
P-MERGE(T, p1, r1, p2, r2, A, p3)
n1 = r1 − p1 + 1
n2 = r2 − p2 + 1
if n1 < n2
        exchange p1 with p2
        exchange r1 with r2
        exchange n1 with n2
if n1 == 0
        return
else
        q1 = ⌊(p1 + r1)/2⌋
        q2 =BINARY-SEARCH(T[q1], T, p2, r2)
        q3 = p3 + (q1 − p1) + (q2 − p2)
        A[q3]=T[q1]
        spawn P-MERGE(T, p1, q1 − 1, p2, q2 − 1, A, p3)
        P-MERGE(T, q1 + 1, r1, q2, r2, A, q3 + 1)
        sync
```

T: Array to be merged
A: Array to save the merged result
p_1, r_1, p_2, r_2: Start and end index of the range to be merged
p_3: The index of the median in A



- $n_1 \geq n_2, n = n_1 + n_2$
- $n_2 = \frac{2n_2}{2} \leq \frac{n_1+n_2}{2} = \frac{n}{2}$
- 最糟的狀況下, x比所有$p_2 \sim r_2$的都大
- 要merge $\left\lfloor \frac{n_1}{2} \right\rfloor + n_2$ 個元素
- $\left\lfloor \frac{n_1}{2} \right\rfloor + n_2 \leq \frac{n_1}{2} + \frac{n_2}{2} + \frac{n_2}{2} = \frac{n_1+n_2}{2} + \frac{n_2}{2} \leq \frac{n}{2} + \frac{n}{4} = \frac{3n}{4}$

Multithreaded Merge

```
P-MERGE(T, p1, r1, p2, r2, A, p3)
n1 = r1 − p1 + 1
n2 = r2 − p2 + 1
if n1 < n2
        exchange p1 with p2
        exchange r1 with r2
        exchange n1 with n2
if n1 == 0
        return
else
        q1 = ⌊(p1 + r1)/2⌋
        q2 =BINARY-SEARCH(T[q1], T, p2, r2)
        q3 = p3 + (q1 − p1) + (q2 − p2)
        A[q3]=T[q1]
        spawn P-MERGE(T, p1, q1 − 1, p2, q2 − 1, A, p3)
        P-MERGE(T, q1 + 1, r1, q2, r2, A, q3 + 1)
        sync
```

$$PM_\infty(n) = PM_\infty\left(\frac{3n}{4}\right) + \Theta(\log n)$$
$$= \Theta(\log^2 n)$$

$PM_1(n) = \Omega(n)$, 因為至少要copy n elements

$PM_1(n) = O(n)$, 見課本p.802

$\Rightarrow PM_1(n) = \Theta(n)$

$\Theta(\log n)$

```
P-MERGE-SORT(A,p,r,B,s)
n=r-p+1
if n==1
     B[s]=A[p]
else let T[1..n] be a new array
```

$q = \left\lfloor \frac{p+r}{2} \right\rfloor$

$q' = q - p + 1$

```
spawn P-MERGE-SORT(A,p,q,T,1)
P-MERGE-SORT(A,q+1,r,T,q'+1)
sync
P-MERGE(T,1,q',q'+1,n,B,s)
```

A: Array to be merged
p,r: Index of the range to be sorted
B: Array to save the result

$$PMS_1(n) = 2PMS_1\left(\frac{n}{2}\right) + PM_1(n) = 2PMS_1\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n \log n)$$

$$PMS_\infty(n) = PMS_\infty\left(\frac{n}{2}\right) + PM_\infty(n) = PMS_\infty\left(\frac{n}{2}\right) + \Theta(\log^2 n) = \Theta(\log^3 n)$$

Parallelism = $\Theta\left(\frac{n \log n}{\log^3 n}\right) = \Theta\left(\frac{n}{\log^2 n}\right)$

Much better now!

怎麼自己學演算法

- 非常重要
- 這堂課沒辦法把所有"重要"的演算法都教完
- 但是希望過程中你已經建立了**自己學習演算法的能力**
- 準備研究所考試的時候可能會需要

我的經驗

- 我也跟著大家一起學習 資料結構 與 演算法
- 一些好方法(自以為)跟大家分享:
  1. 畫圖, 用簡單的例子圖解步驟
  2. 先了解概念(大方向), 不急著看複雜的數學
     (我常說: 如果什麼都沒弄懂, 先弄懂**這個**)
  3. 一次了解**一小部分**就好 (模組化學習), 不急著弄懂所有的部分
  4. **相信**自己一定可以弄懂 (非常重要)
  5. 不知道自己到底懂了沒? 用一些**古怪**的例子來試試 (boundary case)
  6. 練習自己看課本 (非常重要)

期末考內容&型式

- Closed book, 2 x A4 cheat sheets (double-sided)
- 佔學期成績26%
- 包含整學期上課內容, 但以期中考過後的內容為主
- 180 minutes
- 題型與期中考類似 (是非+選擇+解釋)
- Extra office hour ??