CS3383, Winter 2019 Assignment # 8 Sample solutions

Rasoul Shahsavarifar
Faculty of Computer Science, UNB

Questions 1)-Part (a)

$MatMatMult(A_{(n\times n)}, B_{(n\times n)}):\ \ returns\ \ Matrix\ \ C_{(n\times n)}$
Parallel for $i = 1\ \ to\ \ n$
  Parallel for $j = 1\ \ to\ \ n$
    $C[i][j] \leftarrow MatMatMult\_SUBLOOP(A, B, i, j, 1, n)$
return C

$MatMatMult\_SUBLOOP(A, B, i, j, k, k')$
if $k == k'$
  return $a[i][k] * b[k][j]$
else $m = \lfloor (k + k')/2 \rfloor$
$lhalf \leftarrow$ **Spawn**$MatMatMult\_SUBLOOP(A, B, i, j, k, mid)$
  $uhalf \leftarrow MatMatMult\_SUBLOOP(A, B, i, j, mid + 1, k')$
  Sync
  return $lhalf + uhalf$

Analysis: To calculate the $T_1(n)$ of $MatMatMult$, we consider its serialization i.e., by replacing the parallel for loops by ordinary for loops. Therefore, we have $T_1(n) = n^2 T_1'(n)$, where $T_1'(n)$ denotes the work of $MatMatMult\_SUBLOOP$ to compute a given output entry $c[i][j]$. The work of $MatMatMult\_SUBLOOP$ can be obtained by solving the recurrence

$$T_1'(n) = 2T_1'(n/2) + \Theta(1)$$

By applying the first case of the master theorem, we have $T_1'(n) = \Theta(n)$. Therefore, $T_1(n) = \Theta(n^3)$.
To calculate the span (T_∞), we use

$$T_\infty(n) = \Theta(\log n) + \max_{1\le i\le n} iter_\infty(i) + T_\infty(comb.)$$

Note that each iteration of the outer **parallel for** loop does the same amount of work: it calls the inner **parallel for** loop. Similarly, each iteration of the inner **parallel for** loop calls procedure $MatMatMult\_SUBLOOP$ with the same parameters, except for the indices i and j . Because $MatMatMult\_SUBLOOP$ recursively halves the space between its last two parameters (1 and n), does constant-time work in the base case, and spawns one of the recursive calls in parallel with the other, it has span $\Theta(\log n)$. Since each iteration of the inner **parallel for** loop, which has n iterations, has span $\Theta(\log n)$, the inner **parallel for** loop has span $\Theta(\log n)$. By similar logic, the outer **parallel for** loop, and hence procedure $MatMatMult$, has span$\Theta(\log n)$ and the parallelism $\Theta(n^3/\log n)$.

Questions 1)-Part (b) We can efficiently by using the solution in part (a) as a base. We need to replace the upper limits of the nested **parallel for** loops with p and r respectively and we will pass q as the last argument to the call of $MatMatMult\_SUBLOOP$. This subroutine is identical with the one in part (a).

$GeneralMatMatMult(A_{(p\times q)}, B_{(q\times q)}):\ \ returns\ \ Matrix\ \ C_{(p\times r)}$

Parallel for $i = 1\ \ to\ \ p$

  Parallel for $j = 1\ \ to\ \ r$

    $C[i][j] \leftarrow MatMatMult\_SUBLOOP(A, B, i, j, 1, q)$

return C

Analysis: To calculate the work for $GeneralMatMatMult$, we replace the **parallel for** loops with ordinary **for** loops. As before, we can calculate the work of $MatMatMult\_SUBLOOP$ to be $\Theta(q)$ (because the input size to the procedure is q here). Therefore, the work of $GeneralMatMatMult$ is $T_1 = \Theta(pqr)$.
We can analyze the span of $GeneralMatMatMult$ as we did in the part (a), but we must take into account the different number of loop iterations. Each of the p iterations of the outer **parallel for** loop executes the inner **parallel for** loop, and each of the r iterations of the inner **parallel for** loop calls $MatMatMult\_SUBLOOP$, whose span is given by $\Theta(\log q)$. We know that, in general, the span of a **parallel for** loop with n iterations, where the i^{th} iteration has span $iter_\infty$ is given by

$$T_\infty(n) = \Theta(\log n) + \max_{1\leq i\leq n} iter_\infty(i) + T_\infty(comb.)$$

Based on the above observations, we can calculate the span of $GeneralMatMatMult$ as

$$T_\infty = \Theta(\log p) + \Theta(\log q) + \Theta(\log r) = \Theta(\log pqr)$$

The parallelism of the procedure is, therefore, given by $\Theta(pqr/\log pqr)$. To check whether this analysis is consistent with part (a), we note that if $p = q = r = n$, then the parallelism of $GeneralMatMatMult$ would be $\Theta(n^3/\log n^3) = \Theta(n^3/3\log n) = \Theta(n^3/\log n)$.

Questions 1)-Part (c) This part is answered in parts (a) and (b).

Note for TA: Please note that the analysis of the two algorithms are not equally weighted. More weight should be considered for part (b).

Question 2) To compute the transpose of $A_{(n\times n)}$, we give the function $MatTransRec(A, r, c, s)$ to compute the transpose of a $(s \times s)$-sub-matrix starting at a_{rc}. The overall answer (i.e. transpose of A) would be achieved by calling $MatTransRec(A, 1, 1, n)$

$MatTransRec(A_{(n\times n)}, r, c, s):$ *returns transposed* $(s \times s)$*SubMatrix starting at* a_{rc}

if $s == 1$

  return

else

  $s' \leftarrow \lfloor s/2 \rfloor$

  Spawn $MatTransRec(A, r, c, s')$

  Spawn $MatTransRec(A, r + s', c + s', s - s')$

  $SwapMatTransRec(A, r, c + s', r + s', c, s', s - s')$

  Sync

where $SwapMatTransRec$ transposes the $(s_1 \times s_2)$ submatrix starting at $a_{r_1c_1}$ with the $(s_2 \times s_1)$ submatrix starting at $a_{r_2c_2}$

$SwapMatTransRec(A, r_1, c_1, r_2, c_2, s_1, s_2)$.

if $s_1 < s_2$

  $SwapMatTransRec(A, r_2, c_2, r_1, c_1, s_2, s_1)$

else if $s_1 == 1$ //since $s_1 \geq s_2$, must have that s_2 equals 1

  exchange $a_{r_1c_1}$ *with* $a_{r_2c_2}$

else

  $s' \leftarrow \lfloor s_1/2 \rfloor$

  Spawn $SwapMatTransRec(A, r_2, c_2, r_1, c_1, s_2, s')$

  $SwapMatTransRec(A, r_2, c_2 + s', r_1 + s', c_1, s_2, s_1 - s')$

  Sync

As mentioned above, to transpose $A_{(n\times n)}$, we should call $MatTransRec(A, 1, 1, n)$.

Analysis: First, we calculate the work and span of $SwapMatTransRec$ so that we can plug in these values into the work and span calculations of $MatTransRec$. The work $T_1'(N)$ of $SwapMatTransRec$ on an N-element matrix is the running time of its serialization. We have the recurrence

$$T_1'(N) = 2T_1'(N/2) + \Theta(1) = \Theta(N).$$

The span $T_\infty'(N)$ is described by the following recurrence

$$T_\infty'(N) = T_\infty'(N/2) + \Theta(1) = \Theta(\log N).$$

In order to calculate the work of $MatTransRec$, we calculate the running time of its serialization. Let $T_1(N)$ be the work of the algorithm on an N-element matrix, where $N = n^2$, and assume for simplicity that n is an exact power of 2. Because the procedure makes two recursive calls with square submatrices of sizes $(n/2 \times n/2) = N/4$ and because it does $\Theta(n^2) = \Theta(N)$ work to swap all the elements of the other two submatrices of size $(n/2 \times n/2)$, its work is given by the recurrence

$$T_1(N) = 2T_1(N/4) + \Theta(N) = \Theta(N)$$

The two parallel recursive calls in $MatTransRec$ execute on matrices of size $(n/2 \times n/2)$. The span of the procedure is given by maximum of the span of one of these two recursive calls and the $\Theta(\log N)$ span of $SwapMatTransRec$, plus $\Theta(1)$. Since

$$T_\infty(N) = T_\infty(N/4) + \Theta(1) = \Theta(\log N),$$

the span of the recursive call is asymptotically the same as the span of $SwapMatTransRec$, and hence the span of $MatTransRec$ is $\Theta(\log N)$. Thus, $MatTransRec$ has parallelism $\Theta(N/\log N) = \Theta(n^2/\log n^2) = \Theta(n^2/2\log n) = \Theta(n^2/\log n)$.