Assignment #2

Ryan H. Johnston, ide709

NOTE: All sources are at the end of the document

Question #1)

(10 points) Find one of the indices where maximum value occurs in an array A[1..n] of integers in O(1) time on a CRCW Common PRAM model.

(a) Write Pseudocode.

Solution: If duplicate maximum values exist, the highest index will be the one returned.

Algorithm 1 Finding Index of Maximum Value

```
function FindMaxValueIndex(A)
    $n \leftarrow$ length of $A$
    for do $i \leftarrow 0$ to $n-1$, in parallel
        $m[i] \leftarrow TRUE$
    for do $i \leftarrow 0$ to $n-1$ and $j \leftarrow 0$ to $n-1$, in parallel
        if then$A[i] < A[j]$
            $m[i] \leftarrow FALSE$
    for do $i \leftarrow 0$ to $n-1$, in parallel
        if then$m[i] = TRUE$
            max $\leftarrow A[i]$
            max_index $\leftarrow i$
     return max_index
```

(b) Calculate T_p, S_p, E_p, cost, and work of your algorithm.

Solution:

- Sequential time taken, $T_1 = O(n)$, sequential asymptotic runtime
- Parallel time taken, $T_p = O(1)$, parallel asymptotic runtime
- Speedup, $S_p = \frac{T_1}{T_p} = \frac{O(n)}{O(1)} = O(n) \neq O(1)$
- Efficiency, $E_p = \frac{S_p}{p} = \frac{\frac{O(n)}{O(1)}}{p}$
- Cost, $pT_p = p \cdot O(1)$
- Work, $T_1 = O(n + n^2 + n) = O(n^2)$, total operation count across all processes

Question #2)
(15 points) Design an algorithm for multiplying two square matrices of size nx which uses $p \leq n^3$ processors and achieves the fastest parallel execution time of $O(logn)$. You may assume EREW PRAM model.

(a) Give major steps in high level description/pseudocode to answer part (b) - that is, detailed pseudocode is not needed.

Solution: I cannot write pseudocode similar to the solution to problem $1a$, simply because I couldn't design one and prove its correctness. Instead, there exists an algorithm called the DNS algorithm (see the textbook at Ch. 8.2.3) that much smarter people designed and published. The algorithm can multiply two n x n matrices up to n^3 processes and performs multiplication in time of $O(logn)$ by using $\frac{\Omega(n3)}{\log(n)}$ processes. The following is roughly-translated pseudocode for how the algorithm is done.

1. Arrange n^3 processes in a three-dimensional n x n x n logical array.
2. Each process is labeled according to their location in the array.
3. Each labeled process performs a single multiplication that are assigned to a single working process. Mathematically, it can be written as
$$P[i,j,k] = A_ij \cdot B_kj, (0 \leq i,j,k<n) \tag{1}$$
4. Once each process performs a single multplication, as per the previous step, then the contents of
$$C_ij = \sum_{k=0}^{n-1} P[i,j,k]$$
Each element C_ij of the product matrix is obtained by an all-to-one reduction along the k axis, therefore giving the resulting matrix.

(b) For $p \leq n^3$ processors, calculate expressions for T_p, S_p, E_p, cost and work of your algorithm as functions of n and p using O notation.

Solution:

- Number of processors, $p = (n \text{ x } n) \cdot (n \text{ x } n)$, $p \leq n^3$
- Sequential time taken, $T_1 = O(n^3)$, sequential asymptotic runtime

- Parallel time taken, $T_p = O(\log n)$, parallel asymptotic runtime
- Speedup, $S_p = \frac{T_1}{T_p} = \frac{O(n^3)}{O(\log n)} = O(n^3) \neq O(\log n)$
- Efficiency, $E_p = \frac{S_p}{p} = \frac{\frac{O(n^3)}{O(\log n)}}{p}$
- Cost, $pT_p = p \cdot O(\log n)$
- Work, $T_1 = O(n^3 + n^3 + n^3) = O(n^3)$, total operation count across all processes

References

[1] Ananth Grama, Anshul Gupta, George Karypls, Vipin Kumar (1994), *Introduction to Parallel Computing, Second Edition*, The Benjamin/Cummings Publishing Company, Inc. 1994, Pearson Education Limited 2003.

[2] https://www3.nd.edu/ zxu2/acms60212-40212/Lec-07-3.pdf