CSL 857 Model Centric Algorithm Design

Minor 1 , Sem II 2016-17, Max 40, Time 1 1/2 hrs

Note (i) Write your answers neatly and precisely. You won't get a second chance to explain what you have written.
(ii) Every algorithm must be accompanied by proof of correctness and a formal analysis of running time and processor bound. Feel free to quote any result from the lectures without proof - for any anything new, you must prove it first.

1. Consider an array A of n integers and another set of integers $i_1, i_2, \ldots i_k$ where $1 = i_1 < i_j < i_{j+1} < x_k = n+1$. Describe an optimal $O(\log n)$ time PRAM algorithm to compute the partial sums $S_j = \sum_{t=i_j}^{i_{j+1}-1} x_t$ for all $1 \le j \le k-1$.
For example, for inputs 4, 2, 8, 9, -3 and indices 1 , 2, 4, 6 the answer is 4, 2+8 = 10 , 9 -3 = 6. The normal prefix sum can be done with $i_1 = 1, i_2 = n+1$. **(10 marks)**

The main issue here is that of processor allocation - since there are k independent prefix sum problems. These problem sizes can be computed from the indices $i_1, i_2 \ldots$, viz., the size of first problem is $i_2 - i_1$ etc. If a problem size s exceeds $\log n$, then we allocate $\lfloor \frac{s}{\log n} \rfloor$ processors. For $s < \log n$, we will combine these *small* subproblems so that each group will consist of at most $2\log n$ elements. Consider an array $y_1, y_2 \ldots y_k$ such that $y_i < \log n$. Compute prefix sum of y_i and block them in contiguous segments such that each block has a sum bounded by $2\log n$. This can be done by marking the first locations where the sums have exceeded $i \log n$ for $i = 1, 2, \ldots$.

For the *small* subproblems we allocate one processor per block which implies at most $\frac{n}{\log n}$ processors. Each processor takes at most $O(\log n)$ serial time for computation.

For the other subproblems, the total number of processors is

$$\sum_i \lfloor s_i / \log n \rfloor \le \sum_i s_i / \log n = \frac{\sum_i s_i}{\log n} \le n / \log n$$

For each subproblem, the time taken is $O(\log s_i)$ using $s_i / \log s_i$ processors. Since we have $\frac{s_i}{\log n}$ processors, from slow down lemma, we will need $O(\log n)$ time ($\log s_i \times \frac{s_i / \log s_i}{s_i / \log n}$).

2. Consider the following linear recurrence

$$x_i = 3x_{i-1} + x_{i-2} + 5x_{i-3} + 2$$

and $x_1 = 0, x_2 = 2, x_3 = 1$. Design a $O(\log n)$ parallel algorithm to compute x_n for a given n. Do not try to compute an analytical formula. **(15 marks)**
Hint: Write the recurrence as an appropriate matrix.

$$\begin{bmatrix} x_i \\ x_{i-1} \\ x_{i-2} \\ 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 & 5 & 2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{i-1} \\ x_{i-2} \\ x_{i-3} \\ 1 \end{bmatrix}$$

This relation can be expressed as $X_i = A \cdot X_{i-1}$ and $X_0 = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 1 \end{bmatrix}$. From this it follows that $X_i = A^{i-3} \cdot X_0$ and since matrix multiplication is associative, we can use prefix computation with operator 4×4 matrix multiplication to compute the sequence $x_1, x_2 \ldots x_n$.

3. Given two sorted arrays A and B of size n, design an optimal $O(\log n)$ time $n/\log n$ processors algorithm using the following idea.

   For every k-th element in each array, find the cross ranks and then use them to split up the original merging problem into $2\frac{n}{k}$ merging problem of size at most k. Choose an appropriate value of k. **(15 marks)**

   Following the hint, consider every k-th element in the two sorted arrays and denote them by a_{ik} and b_{ik} respectively where $i \leq n/k$. Using one processor per element and using concurrent read find the cross rank of each element in the other array using a binary search. This takes $O(\log n)$ time using $k/\log n$ processors.

   Now consider the cross ranks of two consecutive a_{ik} and $a_{(i+1)k}$ which is given by their positions in the array B - say C_i and C_{i+1}. If they fall within the same k-block of B then we can merge the elements $[a_{ik}, a_{(i+1)k}]$ with $[b_{r_i}, b_{r_{i+1}}]$ that contains at most $2k$ elements. If they fall within different k-blocks of B, then the cross ranks of the elements in B define independent merging problems in $[a_{ik}, a_{(i+1)k}]$ which are again of size at most $2k$.

   To identify the independent merging problems, the crossranks can be used. A subproblem is defined by $a_{ik}, a_{(i+1)k}$ if their cross ranks C_i, C_{i+1} are within the same k-block of B and vice-versa. Allocate one processor to such a pair.

   Each element x in A (B) needs to know its subproblem identity which can be uniquely identified by a pair (a_1, a_2, b_1, b_2). which are the straddling elements in arrays A and B. This can be achieved as follows. When the cross-ranks are determined, we simultaneously mark the elements B_{C_i} and store its index (there may be conflicts when more than one elements have the same cross-ranks, so concurrent write may be necessary and suffices to compute the smallest and the largest index with the same cross-ranks. This would require some extra computations involving cross-ranks). Now an element x needs to find the straddling indices that can be done by an application of prefix computation.

   Each of these merging problems can be completed in $O(k)$ time using 1 processor,and there are at most $2n/k$ subproblems. By choosing $k = \log n$, the entire merging can be completed in $O(\log n)$ parallel time using $n/\log n$) processors.