PARALLEL PROCESSORS APPLIED TO STRING TRANSFORMATIONS

Mohammad Meybodi
Ohio University

Kenneth Williams
Western Michigan University

This work is concerned with the amount of information required, as well as the processing time required, for parallel computing algorithms dealing with problems where the input can be expressed as a character string (or set of strings) and the output is a reordering of the same symbols. Simple examples include string reversal, matrix transposition and sorting. Much more complex problems may be formulated. The approach involves finding a function that maps the position of any symbol in the input string into its corresponding position in the output string. Concurrency can be achieved by simultaneous computation of output positions with different processors.

A classification of problems with related examples based on the amount of information necessary to compute the output function has been partially developed. The output string is a permutation of the input string so, for a given problem incident, function ϕ is a bijection from N to N where N is the set of integers from 1 to n and the input string is of length n.

Consider the problem of reversing a string of characters. The obvious sequential algorithm has complexity O(n). Using the fact that $\phi(\alpha_i) = n - i + 1$, for α_i the i^{th} character in the input string, a parallel computer with P processors can do the reversal in $\lceil n/P \rceil$ time (an O(1) operation if n processors are available).

Define input string $I = \alpha_1\alpha_2 \ldots \alpha_p$ and output string $O = \beta_1\beta_2 \ldots \beta_p$. Each symbol may consist of one or more characters. For each $\alpha \in I$ define $inpos(\alpha) = i$ where $\alpha = \alpha_i$ and $outpos(\alpha) = j$ where $\alpha = \beta_j$. Then Range (ϕ) = {1,...,p}. However, the required Domain(ϕ) may be different for different algorithms. Positions of all the symbols in the output string may be determined in $L + \lceil n/p \rceil \cdot T$ time, for T the time needed to compute one value of ϕ, P the number of processing elements, n the length of the input string and L the amount of time needed to gather the information needed by ϕ. This information gathering time will be different for different problems.

The following cases may be considered, depending on the amount of information necessary.

Case 1. $outpos(\alpha) = \phi(inpos(\alpha))$ only. Example: Identity relationship. $\phi(inpos(\alpha)) = inpos(\alpha)$.

Case 2. $outpos(\alpha) = \phi(inpos(\alpha), n)$ only. Example: String reversal. $\alpha_1\alpha_2 \ldots \alpha_n \rightarrow \alpha_n \ldots \alpha_1$. $\phi(inpos(\alpha), n) = n - inpos(\alpha) + 1$.

Case 3. $outpos(\alpha) = \phi(inpos(\alpha), t_1, t_2, \ldots t_m)$ where the t_i denote fixed information. Example: Matrix transposition, t_1 = number of rows and t_2 = number of columns.

Case 4. $outpos(\alpha) = \phi(\alpha)$ only. Example: Input sequence $\alpha_1\alpha_2 \ldots \alpha_n$ is a random ordering of the integers 1,...,n. For the output we want the numbers in sequential order. Then $outpos(\alpha) = \phi(\alpha) = \alpha$.

Case 5. $outpos(\alpha) = \phi(inpos(\alpha), I)$ only. Examples: Sorting. Infix to prefix conversion (we allow parentheses to map to 0). See [1], [2].

Case 6. $outpos(\alpha) = \phi(inpos(\alpha), t_1, t_2, \ldots, t_m, n)$ only, where the t_i denote fixed information.

Example: Rotate the input string r symbols to the right. $outpos(\alpha) = (inpos(\alpha) + r - 1) \text{MOD}_n + 1$.

Case 7. $outpos(\alpha) = \phi(n, outpos(\beta)) \forall \beta$ with $inpos(\alpha) \neq inpos(\beta)$. Example: The output is a random reordering of the input.

Observation 1 Given n input symbols, each of which receives a unique output index 1,...,n, there are a total of n! different functions which can constitute $outpos(\alpha)$.

Observation 2 Once $outpos(\alpha)$ is known for all α, it may be applied in $O(\lceil n/P \rceil)$ time where P processors are available.

References

[1] E. Dekel and S. Sahni, "Parallel Generation of Postfix and Tree Forms", ACM Transactions on Programming Languages and Systems, Vol 5, No 3, July, 1983, pp. 300-317.

[2] D.E. Muller and F.P. Preparata, "Bounds to Complexities of Networks for Sorting and for Switching", Journal of ACM, 1975, pp. 195-201.