December 1988 Report No. STAN-CS-88-1240

On Separating the EREW and CREW PRAM Models

by

E. Gafni, J. Naor, and P. Ragde

Department of Computer Science

Stanford University

Stanford, California 94305

On Separating the EREW and CREW PRAM Models

Eli Gafni * Joseph Naor † Prabhakar Ragde ‡

Abstract

In [6], Snir proposed the Selection Problem (searching in a sorted table) to show that the CREW PRAM is strictly more powerful than the EREW PRAM. This problem defines a partial function, that is, one that is defined only on a restricted set of inputs. Recognizing whether an arbitrary input belongs to this restricted set is hard for both CREW and EREW PRAMs. The existence of a total function that exhibits the power of the CREW model over the EREW model was an open problem. Here we solve this problem by generalizing the Selection problem to a Decision Tree problem which is defined on a full domain and to which Snir's lower bound applies.

*Department of Computer Science, University of California, Los Angeles, CA 90024

†Department of Computer Science, Stanford University, Stanford, CA 943052140. Supported by contract ONR N00014-88-K-0166.

‡ Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. N2L 3G 1

1 Introduction

Consider the Selection problem, which we denote (S): given an input vector X = $\langle x_1, x_2, \ldots, x_{n-1} \rangle$ and an input y, where all inputs are integers, find the index i such that $x_i < y \leq x_{i+1}$. (By definition $x_0 = -\infty$ and $x_{n+1} = \infty$). Problem (S) is just the problem of searching in a sorted table of integers.

Snir [6] considered this problem in the context of parallel computation in two different PRAM models. A PRAM consists of a set of processors $P_0, P_1, \ldots$ which communicate by means of cells $M_0, M_1, \ldots$ of shared memory. One step of computation consists of three phases. In the read phase, each processor may choose one cell to read from. In the compute phase, an arbitrary amount of local computation can take place. In the write phase, each processor may choose one cell to write into. The models that Snir considered differ in the degree of simultaneous access to shared memory that is allowed. In the EREW PRAM, no two processors may simultaneously read or write into the same cell. In the CREW PRAM, simultaneous read access is permitted, but not simultaneous write access.

It is easy to see that the complexity of problem (S) on a CREW PRAM is $O(1)$, and Snir [6] proved an $\Theta(\sqrt{\log n})$ upper and lower bound on solving the problem in the EREW model. His proof proceeds by using Ramsey's Theorem to restrict the set of inputs so that the behaviour of an algorithm solving the problem depends only on the relative order of the input values. Essentially, processors may only make comparisons and gather input values, and an information-theoretic argument shows that this cannot be done quickly. The use of Ramsey's Theorem means that the lower bound holds only if the input numbers are drawn from a large enough range.

A more serious problem with this lower bound proof than the size of the range needed is that the problem is only defined on a restricted set of inputs (termed a *cleft domain* in [4]). The problem of testing whether the input is valid (that is, the x's are sorted) requires $\Omega(\log n)$ time in the CREW model. (This follows from the lower bound of [I] on the computation of the OR of 12 bits). It could be argued that knowing that the input is of a special form gives information to the CREW PRAM that the EREW PRAM cannot use, and thus the comparison is "unfair". Examples have been given of PRAM models which can be separated by the use of functions defined on partial domains, but which are equal or incomparable when considering functions on full domains ([2], [3]).

In the next section we show how the Selection problem (S) can be reformulated as a. Decision Tree problem, such that the output is well defined for any input.

2 Generalization of the Selection Problem to a Decision Tree Problem

Let ***T*** be a complete rooted binary tree of size n such that $n = 2^h - 1$ where h is an integer. An input variable is associated with each node of ***T.*** The variable $x_{n/2}$ is associated with the root, $x_{n/4}$ and $x_{3n/4}$ with the left and right child of the root respectively, and so on. More precisely, if a node has x_i associated with it, and i = $(2k + 1)2^b$, then the left child of the node has x_j associated with it, and the right child has x_k associated with it, where j = $(2k - \frac{1}{2})2^b$ and $k = (2k + \frac{1}{2})2^b$. We number the nodes, giving a. node the same index as the variable associated with it.

We now state the Decision Tree problem, denoted problem (D): A path from each node to one of the leaves is defined inductively. The successor of internal node i is the left child of i if $y < x_i$ and the right child of i if $y \geq x_i$. There is a. unique root-leaf path terminating at some leaf j. The output of the problem is j $-$ 1 if $y < x_i$ and j if $y \geq x_i$.

Theorem 2. 1 *Problem (S)* ***can be solved in*** $\Theta(\log \log n)$ ***time*** **in** ***the CREW model.***

Proof: Problem (S) is solved in the CREW model by using the "path doubling" technique. A processor P_i is associated with each node i in the tree. P_i reads y and x_i, thereby determining the successor of node i. This information is stored in memory, say in location i of array S. For a leaf j, let S(j) = j. Then, in parallel, each processor P_i executes the instruction S(i) $\leftarrow$ S(***S(i)),*** a. total of log log **n** times. After this is done, $S(n/2)$ = j means that node j is the leaf at the end of the path from the root. In $O(1)$ steps the answer can be computed.

To sec that a. CREW PṘAM requires $\Omega(\log \log n)$ time to solve problem (D), we invoke a. result of Simon [5], which states that any nondegenerate Boolean function on n variables requires $\Omega(\log \log n)$ steps to compute on CREW. Our problem does not define a. Boolean function, since inputs are tuples of integers, but we can construct a BOO1ea.n function g by letting $y = 1$, restricting $x_1, x_2, \ldots x_{n-1}$ lo have value O-or 1, and defining the output of g to be f (mod 2). The resulting g is at least, as easy to compute as f, and is a nondegenerate Boolean function of $n - 1$ variables.

Theorem *2.2 Problem (D) requires* $\Theta(\sqrt{\log n})$ *time to solve in the EREW model.*

Proof: The purpose of demonstrating an $0(\sqrt{\log n})$ algorithm is to show that the lower bound of Snir is the best possible, as the lower bound model does not charge for local computation. As before, a processor P_i is associated with node i. In the first step of the algorithm, P_i reads x_i and stores this value in node i. We note that in O(1) steps a processor at a node can read any information stored in its left and right children and coalesce this information along with any information it has. Thus, in $O(\sqrt{\log n})$ steps, a node v that is at level $k\sqrt{\log n}$ + **1** for some integer ***k*** can gather the values of all variables associated with nodes in the subtree of height $\sqrt{\log n}$ below node v. Knowing these values and the value of y, a processor can determine in one step the node that is $\sqrt{\log n}$ levels below v on the path from v. In effect, the binary tree has been compressed so that it is now a tree of height $\sqrt{\log n}$ and fanout $2^{\sqrt{\log n}}$. The naive sequential algorithm to find the bottom of the root-leaf path can now be run, taking $O(\sqrt{\log n})$ steps.

To prove the lower bound, we show that problem (S) is reducible to problem (D) in time 0(**1).** In fact, problem (S) is just problem (D) restricted to inputs in which the x's are sorted. The root-leaf path defined by problem (D) is just the sequence of variables that would be queried by binary search.

Another way of stating the separation implied by the previous two theorems is that for each integer ***T,*** there exists a problem which can be solved in ***T*** steps in the CREW model, but which requires $2^{\Omega(T)}$ steps on the EREW model.

3 Separations on Boolean input and output

These results can be extended slightly to show a. lower bound for a problem with integer input but Boolean output. The problem is just problem (D), but the output is taken to be the output of problem (D) mod 2. To see that Snir's lower bound applies to this problem, one must examine Snir's proof. He shows that if $o(\sqrt{\log n})$ steps are used by some algorithm, there exist two inputs in the restricted domain and an integer i such that the outputs of problem (S) on those two inputs are i and i + **1** respectively, and the computation of the EREW PRAM on the two inputs is identical. For two such inputs, the output of problem (D) mod 2 would also differ, and the lower bound follows.

In [6], Snir gives a lower bound for a problem with Boolean input and output. The problem is to identify the switching index when the input is a string of O's followed by a string of 1's. A lower bound of $\Omega(\log(n/p))$ time in the EREW model is proven, where p is the number of processors. The problem can be solved in $O(\log n/logp)$ time in the CREW model.

In the same vein as in the previous section, it is easy to see that if we modify problem (D) to restrict the inputs to being Boolean, and further fix $y = 1$, then Snir's problem is just the modified problem defined on a restricted set of inputs. Thus the modified problem (D) takes time at least $\Omega(\log(n/p))$ time in the EREW model. Problem (D) can be solved in time 0((log $n/logp$) log logp) in the CREW model. The p processors are assigned to nodes in the first log p levels of the tree and in O(log ***logp)*** steps can find out which node at the lowest level is reached by the root-leaf path. This procedure is then repeated log $n/$ log p times until the bottom of the tree is reached.

References

[1] S. Cook, R. Reischuk and C. Dwork, Upper and lower bounds for parallel random access machines without. simultaneous writes, *SIAM J. Computing*, 15 (1986) 87-97.

[2] F.E. Fich, P. Ragde, and A. Wigderson, ***Relations between*** *concurrent-write* ***models of*** *parallel* ***computation, SIAM J.*** *Computing* ***17*** (1988) ***606-627.***

[3] V. Grolmusz and P. Ragde, *Incomparability* ***in parallel*** *computation*, Proc. 2Sth IEEE Symposium on Foundations of Computer Science, 1 987.

[4] R. Reischuk, Simultaneous writes do not help to compute simple arithmetic functions, *SIAM J. Computing* 16 (1987).

[5] H.-U. Simon, A tight $\Omega(\log \log n)$ bound on the time for parallel RAMs to compute nondegenerated Boolean functions, *Information* ***and*** *Control* 55 (1982) 102-107.

[6] M. Snir, On parallel searching, *SIAM* ***J.*** *Computing* **14** (1985) 688-708.