TR80-013

On the Complexity of Vector Computations in Binary Tree Machines

D. M. Tolle
W. E. Siddall

January 1981

On the Complexity of Vector Computations in Binary Tree Machines

D. M. Tolle

W. E. Siddall*

University of North Carolina at Chapel Hill

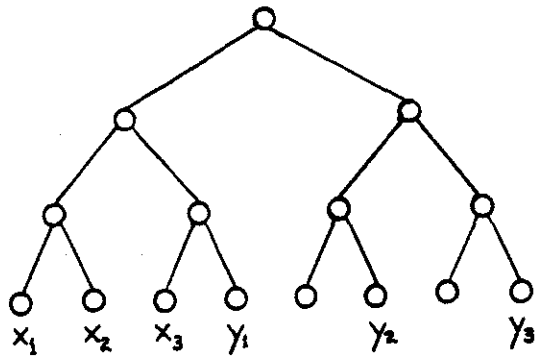
Computational complexity, analysis of algorithms, parallel processing

This paper establishes upper and lower bounds for the time required to perform certain vector operations in a binary tree machine of the kind introduced by Magó[1]. (Tolle[2] proposes another such machine.) This paper also characterizes the space-time tradeoffs available in such machines for certain vector operations.

The machines considered here consist of a complete binary tree of "cells," each of which consists of a processor and a

* W. E. Siddall's contribution to this work was supported by National Science Foundation grant MCS-7802778.

small amount of memory. Let two n-vectors $x = \langle x[1], \ldots, x[n] \rangle$ and $y = \langle y[1], \ldots, y[n] \rangle$ of atomic symbols ("atoms") be stored in the leaf cells of the tree, in left-to-right order, with at most one atom per cell, and with vector x lying entirely to the left of vector y. (The atoms might be floating point numbers, for instance.) Here is a small example:



Let a permutation q be defined on the set 1, ..., n. Consider the problem of moving the elements of the two vectors along the arcs of the tree so that, for each i, x[i] meets y[q(i)] in some cell of the tree. This is a necessary step in computing any element-by-element combination of x and y, such as the inner product. Assume that each arc of the tree is a two-way channel capable of moving one atom (and an associated subscript) between its two cells in each direction in one unit of time. We call the problem of bringing x[i] together with y[q(i)] for all i, $1 \leq i \leq n$, the n-vector matching problem or the problem of bringing two

n-vectors together. We are interested here in the amount of time needed to solve this problem, for various permutations g. We will see that the time required depends upon the initial distribution of vector elements in the leaf cells of the tree. One important aspect of the distribution is the amount of space used by the two vectors: the number of leaf cells between and including the leaf cells occupied by x[1] and y[n]. We assume that there is some means by which each cell can determine, at each time step, which of its arriving atoms should next be sent on, and along which arcs they should be sent. This assumption is easily satisfied for the most commonly encountered permutations, such as the identity and the reversal permutations.

Notation. The two vectors x and y have n elements each. All the logarithms in this paper are base 2. Let f(n) and g(n) be functions of some integer quantity n. Then we say that f(n) is O(g(n)) if there is some positive constant c and some integer n0 such that $|f(n)| \leq c*|g(n)|$ for all $n \geq n0$. We say that f(n) is $\Theta(g(n))$ if there are positive constants c1 and c2 such that $c1*g(n) \leq f(n) \leq c2*g(n)$ for all sufficiently large values of n. Let h denote the height of the machine tree. Notice that if n is Θ(number_of_leaf_cells), then h is $\Theta(\log(n))$. We say that a cell sees an element of a vector if the element initially lies in the subtree of which that cell is the root.

Proposition 0. Any n-vector matching can be done in O(n+h) time.

Proof: Let the elements of x move up to the root cell of the tree and then be broadcast downward to all the leaf cells of the tree. It takes h time steps for the first element of x to reach the root, another n-1 steps for the last element of x to reach the root, and another h steps for the last element of x to reach the leaf cells. (Notice that if h is O(n), which is usually a reasonable assumption, then any n-vector matching can be done in O(n) time. Kehs[3, pp. 140-144] has shown that if additional arcs are inserted in the tree, connecting each cell with its two horizontal neighbors, then any n-vector matching can be done in sub-linear time.)

Proposition 1. For the identity permutation, the n-vector matching problem requires at least $\Theta(n)$ time.

Proof: Consider the lowest cell, A, that sees at least half of each vector. We show that at least half the pairs (x[i],y[i]) are "split" by A, in the sense that at least one element of the pair must travel through (or to) A in order to meet its partner. To show that a pair is split by a cell, it suffices to show that one element of the pair lies in one subtree of the cell and the other element of the pair lies either in the other subtree of the cell or outside the

two subtrees of the cell. We consider two cases:

Case 1. Assume that x[n] lies in A's left subtree. Then A's left son must see at least half of x, and therefore cannot see as much as half of y (by the definition of A). Thus, the right half (at least) of y lies to the right of A's left subtree. We show that A splits each element of the right half of y from its partner. The elements of y that lie in A's right subtree are certainly split from their partners by A, since no element of x lies in A's right subtree. The part of y that lies to the right of A's right subtree must constitute no more than half of y (else A could not see at least half of y). Thus, the partners of the elements in that part of y all lie in A's left subtree; thus, the elements of that part of y are are also split by A from their partners. Hence A splits each element of the right half of y from its partner.

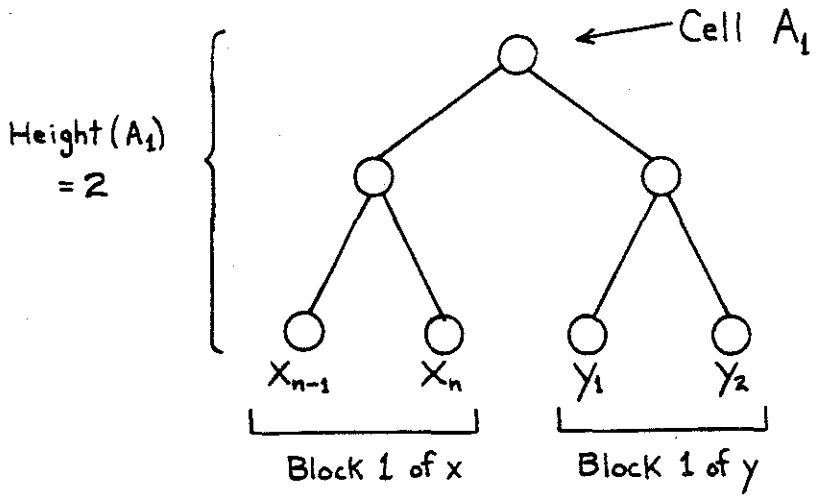
Case 2. Assume that x[n] lies in A's right subtree. Then y[1] lies in A's right subtree, and an argument symmetrical to the one above shows that at least half the elements of x lie to the left of A's right subtree and are split by A from their partners in y.

Thus, in either case, at least n/2 elements must travel to or through A in order for the vectors to be brought together. This takes at least n/2 time units, which is

$\Theta(n)$.

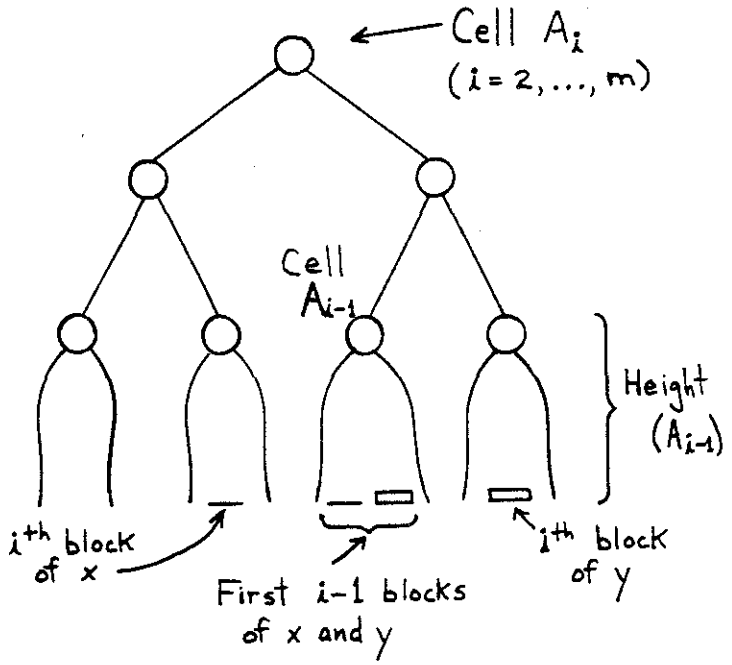
Proposition 2. Some n-vector matchings (including reversal) can be done in $\Theta(\text{sqrt}(n))$ time, using $\Theta(4^{**}\text{sqrt}(n))$ space.

Proof: Assume for simplicity that $n = m(m+1)$ and that $m = 2^{**}k$, for some integer k. Then m is $\Theta(\text{sqrt}(n))$. We arrange each vector in m blocks. Let the blocks of x be indexed from right to left and those of y be indexed from left to right. For each vector, let the i-th block ($i = 1, \ldots, m$) contain 2i elements. Assume that the permutation matches up (in some order) the elements of the i-th block of y with the i-th block of x. (Notice that the reversal permutation satisfies this assumption.) Let the first block of x and the first block of y lie in adjacent subtrees of height 1:



Then these two blocks can be brought together at node A[1], in 3 units of time.

For each of the remaining m-1 pairs of blocks, assume that the arrangement is this:



It is clear that the i-th block of y can be brought together with the i-th block of x at cell A[i], using paths in the tree not used by earlier blocks, in time:

$$\begin{aligned} \text{Time(i-th block)} &= 2i + \text{height}(A[i]) - 1 \\ &= 2i + \text{height}(A[1]) + 2(i-1) - 1 \\ &= 4i - 1 \end{aligned}$$

The maximum occurs for i = m, and thus the time taken for the entire n-vector matching is:

Time(m-th block) = 4m - 1 < 4*sqrt(n) -1

which is Θ(sqrt(n)).

The space used by the vectors is

$$\begin{aligned} & 4m + 2^{**}(\text{height}(A[m-1])) \\ &= 4m + 2^{**}(\text{height}(A[1]) + 2(m-2)) \\ &= 4m + 2^{**}(2m - 2) \\ &< 4*\text{sqrt}(n) + (4^{**}\text{sqrt}(n))/4 \end{aligned}$$

which is Θ(4**sqrt(n)).

Proposition 3. Every n-vector matching takes at least sqrt(n) time.

Proof: Let LCA[i] (Lowest Common Ancestor of i) denote the (unique) cell of minimum height that sees both x[i] and its partner y[g(i)]. Then x[i] is in the left subtree and y[g(i)] is in the right subtree of LCA[i]. Notice that a given cell in the machine may serve as LCA[i] for more than one value of i. Let d denote the number of distinct cells that serve as LCA[i] for one or more values of i. No two of these d cells can be at the same height, because two distinct cells at the same height have disjoint subtrees and therefore cannot both see elements of both x and y. For any cell c in the machine, let #LCA(c) denote the number of

values of i for which c serves as LCA[i]. Then d ≥ n/max(#LCA(c)), where the max is taken over all the cells of the machine. Since either x[i] or y[q(i)] (or both) must travel through (or to) LCA[i], it is clear that the time required for the matching is at least max(#LCA(c)). It is also apparent that the time required is at least max(height(LCA[i])), taken over i = 1, ...,n.

Now, assume that a given n-vector matching can be done in time t(n). Then max(#LCA(c)) ≤ t(n), so d ≥ n/t(n). Since all d of the cells serving as LCA[i]'s must be at different heights, the highest of them must have height at least n/t(n), so the time required by the matching is at least n/t(n). That is, t(n) ≥ n/t(n). Thus t(n)*t(n) ≥ n, so t(n) ≥ sqrt(n). (Notice that the space required is at least (1/4)*2**(n/t(n)).)

Proposition 4. If the vectors are constrained to lie within c*n**p space, for any constants c > 0 and p ≥ 1, then every n-vector matching takes at least Θ(n/log(n)) time.

Proof: As was noted in the proof of Proposition 3, the space needed to perform an n-vector matching in time t(n) is at least (1/4)*2**(n/t(n)). If the space is no more than c*n**p, then 4*c*n**p ≥ 2**(n/t(n)). Taking logarithms of both sides, we see that 2+log(c)+p*log(n) ≥ n/t(n), so that t(n) ≥ n/(2+log(c)+p*log(n)), which is Θ(n/log(n)).

Proposition 5. If the vectors are allowed to use n**p space, for any constant p > 1, then there are some distributions of the vector elements for which some n-vector matchings (including reversal) can be done in $\Theta(n/\log(n))$ time.

Sketch of Proof: Given p > 1, choose k = 2/(p-1). As in the proof of Proposition 2, break the x and y vectors into blocks of elements, but let each block be of size approximately s = kn/log(n). Then each vector has roughly n/(kn/log(n)) = log(n)/k blocks. Arrange the blocks as in the proof of Proposition 2. Letting m denote the number of blocks, we see that the time required is:

s + height(A[m]) - 1, which is roughly

kn/log(n) + 2(log(n)/k - 1) + log(kn/log(n))

= kn/log(n) + (1+2/k)log(n) + log(k) - 2 - log(log(n)),

which is $\Theta(kn/\log(n)) = \Theta(2n/((p-1)\log(n))) = \Theta(n/\log(n))$. The amount of space used is bounded above by 2**height(A[m]), which, for sufficiently large n, is no more than 2**((1+2/k)log(n)), which is n**p.

(A rigorous proof, using the ceiling function, is straightforward but tedious)

Proposition 6. If the vectors are constrained to lie within c*n space, for some constant $c \geq 2$, then every n-vector matching requires at least $\Theta(n)$ time.

Proof: Let R[1] denote the lowest cell that sees both x[n] and y[1]; x[n] must be in its left subtree, y[1] in its right. Consider the sequence R[1], R[2], ..., of right ancestor cells of R[1]: those ancestors of R[1] that have R[1] in their left subtree. Suppose that R[k] is the lowest right ancestor cell of R[1] that sees all of y. (k may be 1.)

Every element of y lies in the right subtree of exactly one R[i]. Since no element of x lies in the right subtree of any R[i], any element of y that lies in the right subtree of R[i] is split by R[i] from its partner in x. Let C denote ceiling(log(c)). If $k \leq 2+C$, then some R[i] must split at least n/(2+C) elements of y from their partners in x.

If $k > 2+C$, consider the penultimate 1+C of the R[i]: R[k-C-1], R[k-C], ..., R[k-1]. Each of these sees at least twice as many of the cells between y[1] and y[n] (inclusive) as its predecessor does. Thus, R[k-1] sees at least $s*2^{**}(C+1)$ of these cells, where s is the number of them seen by R[k-C-2]. Since y uses no more than c*n cells, we have:

$2*c*s \leq s*2**(C+1) \leq c*n,$

and thus $s \leq n/2$. This implies that the number of elements of y seen by the first k-C-2 cells of the right-ancestor sequence is no more than n/2, and thus that the number of elements of y seen by the last C+2 cells in their right subtrees is at least n/2. Thus, some R[i] splits at least $n/(2*(C+2))$ pairs of elements. Hence, the time required to do the matching is at least $\Theta(n)$.

Summary

For the class of binary tree machines considered here, two disjoint n-vectors, stored with at most one vector element per cell in the leaf cells of the tree, can be brought together (matched) element by element, according to any permutation, in O(n+h) time, where h is the height of the tree. If the space occupied by the two vectors (including any interspersed empty leaf cells) is only linear, then at least linear time is required to bring them together, regardless of the permutation. If the vectors occupy polynomial space, then at least $\Theta(n/\log(n))$ time is required. Some matchings (such as reversal) can be done in $\Theta(n/\log(n))$ time if the vectors are allowed to occupy n**p space, for any p > 1. Some matchings (such as reversal) can be done in $\Theta(\mathrm{sqrt}(n))$ time if the vectors are allowed to occupy $\Theta(4**\mathrm{sqrt}(n))$ space. However, no matching can be

done in less than sqrt(n) time, and some matchings (such as identity) always require at least linear time, regardless of the amount of space used by the vectors.

Acknowledgements

Thanks to Hollins Williams, who conjectured something akin to Proposition 1. Thanks also to Don Stanat, for asking whether all n-vector matchings require $\Theta(n)$ time in these machines, and for helpful suggestions concerning the paper. Special thanks to Gyula Magó, for inventing the machine that makes these questions interesting. Further thanks to all the above and to Anne Presnell, Roy Pargas, Lee Nackman, and Vicki Baker for their comments on this paper.

References

1. Magó, Gyula A. "A network of microprocessors to execute reduction languages." Two parts. International Journal of Computer and Information Sciences 8, 5 (October 1979) and 8, 6 (December 1979).

2. Tolle, D. M. "Coordination of computation in a binary tree of processors: a machine design." Ph.D. dissertation, Department of Computer Science, University of North Carolina at Chapel Hill. In preparation.

3. Kens, David R. "A routing network for a machine to execute reduction languages." Ph.D. dissertation, Department of Computer Science, University of North Carolina at Chapel Hill, 1978.