ZBORNIK RADOVA
Prirodno-matematičkog fakulteta
Univerziteta u Novom Sadu
Serija za matematiku, 17,1(1987)

REVIEW OF RESEARCH
Faculty of Science
University of Novi Sad
Mathematics Series, 17,1(1987)

PARALLEL ALGORITHMS FOR FINDING THE MEASURE OF THE UNION OF INTERVALS

Ivan Stojmenović and Lidija Čomić

University of Novi Sad, Faculty of Science,
Institute of Mathematics, Dr. I. Djuričića 4,
21000 Novi Sad, Yugoslavia

ABSTRACT

The paper presents an algorithm for finding the measure of the union of intervals. The algorithm is based on a divide-and-conquer approach. When implemented on a sequential computer it runs in optimal O(n*log(n)) time. The algorithm is modified for several parallel models of computations. The running time of the algorithm on mesh-connected parallel computers is optimal O(n**(1/2)), while the running times on a concurrent read exclusive write PRAM and concurrent read concurrent write PRAM are O(log**2(n)) and O(log(n)) respectively, which is very efficient.

1. INTRODUCTION

One of problems that has raised considerable interest and can be traced as the origin of many problems in rectilinear

---

AMS Mathematics Subject Classification (1980): Primary 68U05
Secondary 68Q10.
Key works and phrases: Measure of union of intervals, parallel computation, mesh-connected computer, CREW PRAM, CRCW PRAM, computational geometry.

geometry is the following:

Given a set of n intervals [a(1),b(1)], ..., [a(n), b(n)] in the real line, it is desired to find the measure of their union, i.e. the measure of the part of the line covered by at least one interval.

Klee [4] has given an O(n*log(n)) sequential solution to the problem which has been proved to be optimal (cf. [11]). Parallelizing the algorithm is a rather difficult task.

We shall present a divide-and-conquer solution to the same problem. This algorithm has the same sequential running time O(n*log(n)).

Our algorithm can be easily implemented on several different parallel models of computations. We shall consider two theoretical and one practical model of parallel computers.

The first model we shall consider is the computational model with a shared memory providing constant communication time between any two processors, in which concurrent reads are allowed, but no two processors should attempt to simultaneously write in the same memory location. We shall henceforth refer to this model as the CREW PRAM (concurrent read exclusive write parallel random access machine).

The second model we shall consider is the CRCW PRAM (concurrent read concurrent write PRAM) in which several processors can simultaneously attempt to write in the same memory location, and only the maximum of values to be written is stored.

We shall also consider a practical model of computation - mesh-connected parallel computer.

A mesh-connected parallel computer of size n is a set of n synchronized processing elements (PEs) arranged in an n**(1/2) x n**(1/2) grid. Each PE is connected via bi-directional unit-time communication links to its four neighbours, if they exist.

Each processor has a fixed number of registers and can perform standard arithmetic and comparisons in constant time. It can also send the contents of a register to a neighbour and receive a value from a neighbour in a designated register in O(1) time units. Each PE in the leftmost column has an I/O port. Thus, we can "load" S in O(n**(1/2)) time units such that each processor contains exactly one arbitrary point of S. Each PE contains a unique identification register (ID), the contents of which correspond to that PE's snake-like index.

We shall use standard MCC data movement operations: rotating data within a row (column), sorting, compression of data, Random Access Read (RAR) and Random Access Write (RAW) (see [8,9,10]).

For the algorithm presented in this paper, we shall assume that there exists initially n or fewer intervals, distributed one interval (two endpoints) per processor on a parallel computer of size n.

To simplify the exposition of our algorithms we shall assume that n=4**k for some integer k and all points have distinct coordinates.

Sorting algorithms [2,5,13] are basic algorithms which provide an efficient (parallel) solution to some geometric problems. Sequential solutions to computational geometry problems are surveyed in [11] while parallel solutions for some of these problems on different models of computations are given in [1,3,6,7,8,9] and some other papers.

Our algorithm for finding the measure of the union of intervals is optimal on mesh-connected computers and efficient on CREW PRAM and CRCW PRAM.

2. ALGORITHM FOR FINDING THE MEASURE OF THE UNION OF INTERVALS

The algorithm works in the following way:

Preprocessing step: Sort the endpoints a(1),b(1),..., a(n),b(n) of intervals.

Procedure Measure-of-union-of-intervals (S,n)

Step 1. Find the median point m of the given 2n points.

Step 2. Find all intervals [a(i),b(i)] containing the median point m. Let l be the smallest left endpoint a(i) among intervals containing m. Similarly we compute r as the greatest right endpoint of these intervals. The number of these "middle" intervals we denote by p.

Step 3. Let L be the set of intervals that are completely on the left of m and R be the corresponding set of intervals that are on the right of m. Both L and R contain (n-p)/2 intervals. We "cut" the intervals from L and R by modifying the right endpoint b(i) of each interval [a(i),b(i)] from L so that b(i):=min(b(i),l) and by modifying the left endpoint a(i) of each inverval [a(i),b(i)] from R so that a(i):= =max(a(i),r). We discard T(l) intervals from L for which a(i)>=l is satisfied (similarly for "covered" T(r) intervals from R for which b(i)<=r) is valid). Let k(l) and k(r) be the indexes of the points l and r in the sorted list of interval's endpoints respectively. Then all endpoints b(i) of intervals from L with b(i)>=l receive new rank in the sorted list of S, which is between k(l) and (n-p)/2-T(l)(We can store the latter two numbers instead of the exact rank.) We can do the same for the set R. Note that the median point m of the set L cannot be one with the rank between k(l) and (n-p)/2-T(l) (i.e. one of the modified endpoints) because we have discarded "covered" intervals.

Step 4. Recursively find the measures l' and r' of modified intervals for each of sets L and R separately (in parallel).

Step 5. The measure of the union of intervals is l'+r'+r-l.

All operations in the given algorithm are easily implemented on the parallel models of computation we are considering. The only nontrivial operation is computing the number p on mesh-connected computers.

One can compute the number p on a mesh in the following way: intervals containing m store 1 in a register and the remaining ones store 0. Now the sum of all elements can be calculated first by computing the sum of the value of registers in PEs in each row and storing it in the leftmost PE of each row. Next, the sum of values of registers in PEs in the leftmost column is calculated.

The proof of the correctness of the algorithm is straightforward.

The running time T(n) of the above algorithm can be expressed as:

1) on a sequential computer
$T(n)=2T(n/2)+O(n)$, i.e. $T(n)=O(n*\log(n))$;
2) on a mesh-connected computer
$T(n)=T(n/2)+O$, i.e. $T(n) = O(n^{**}(1/2))$
3) on a CREW PRAM
$T(n)=T(n/2)+O(\log(n))$, i.e. $T(n)=O(\log^{**}2(n))$;
4) on a CRCW PRAM
$T(n)=T(n/2)+O(1)$, i.e. $T(n)=O(\log(n))$.

3. CONCLUSION

The given algorithm can also solve the following problem: Given n intervals in the real line, determine if there are two which intersect. Obviously, there is no intersection of intervals iff the measure of the union of intervals is equal to the union of measures of the intervals (cf. [11]).

Our problem is a dimensional specification of the problems of finding the measure of area, perimeter and contour of n isothetic rectangles, i.e. rectangles with sides parallel to the coordinate axes.

There are two solutions to the problem of computing the total area covered by isothetic rectangles [9,6], both for mesh-connected computers. Owing to our specification, we have given a simpler solution of our problem using a different idea.

Solving the mentioned rectangle problems on various models of parallel computation is now a problem for further investigation.

REFERENCES

[1] Aggarval,A., Chazelle,B., Gulbas,L., O'Dunlaing,C., Yap,C., *Parallel Computational Geometry*, IEEE Symp.Par.Process, 1985, 468-477.

[2] Ajtai,M., Komlos,J., Szemeredi,E., *An O(n*log(n)) Sorting Network*, Combinatorica 3, 1, 1983, 1-19.

[3] Atallah,M.J., Goodrich,M.T., *Efficient Parallel Solutions to Geometric Problems*, IEEE Symp.Par.Process., 1985, 411-417.

[4] Klee,V., *Can the Measure of U[a(i),b(i)] Be Computed in Less than O(n*log(n)) Steps?*, Amer.Math.Monthly, 84, 4, 284-285, 1977.

[5] Leighton,T., *Tight Bounds on the Complexity of Parallel Sorting*, IEEE Trans.Comput., C-34, No.4, 1985, 344--354.

[6] Lu,M., Varman,P., *Mesh-Connected Computer Algorithms for Rectangle-Intersection Problems*, IEEE Int.Conf. on Parallel. Proc. 1986, 301-307.

[7] Lu,M., Varman,P., *Solving Geometric Proximity Problems on Mesh-Connected Computers*, Proc. IEEE 1985 Workshop on Computer Architecture for Pattern Analysis and Image Database Management, Miami Beach, pp.248-255.

[8] Miller,R., Stout,Q.F., *Computational Geometry on a Mesh--Connected Computer*, Proc. 1984, IEEE Int.Conf. on Parallel Processing, 66-73.

[9] Miller,R., Stout,Q.F., *Mesh Computer Algorithms for Computational Geometry*, Technical Report 86-18, Dept. of Comp.Sci., State Univ. of New York at Buffalo, July 1986, pp. 50.

[10] Nassimi,D., Sahni,S., *Finding Connected Components and Connected Ones on a Mesh-Connected Parallel Computer*, SIAM J. Computing, 9, 1980, 744-757.

[11] Preparata,F.P., Shamos,M.I., *Computational Geometry, An Introduction*, Springer-Verlag, New York, 1985.

[12] Stout,Q.F., *Broadcasting in Mesh-Connected Computers*, Proc. 1982 Conf. on Info. Sciences and Systems, Princeton Univ., pp. 85-90.

[13] Thompson, C.D., Kung, H.T., *Sorting on a Mesh-Connected Parallel Computer*, Comm. ACM 20(4) (1977) 263-271.

REZIME

PARALELNI ALGORITMI ZA NALAŽENJE MERE UNIJE INTERVALA

U radu je prikazan jedan algoritam za nalaženje mere unije intervala. Algoritam je zasnovan na principu podele i spajanja rešenja. Naasekvencijalnom kompjuteru dati algoritam se izvršava u optimalnom O(n*log(n)) broju operacija. Vreme rada algoritma na mrežno-povezanom paralelnom kompjuteru je optimalno O(n*log(n)), a na paralelnim modelima CREW PRAM i CRCW PRAM je efikasno (O(log**2(n)), odnosno O(log(n))).

Received by the editors December 16, 1986.