ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 3, March 2015

A Study on Performance Analysis of Different Parallel algorithms Implemented for Geometric Problems

Kalyan Kumar Jena

Assistant Professor, Department of CSEA, IGIT, Sarang , Odisha , India

Abstract: There are different techniques to make parallel programs. In parallel programming, independent pieces of tasks that may run in parallel. The objective of parallel algorithm design is to develop parallel computational methods that run very fast with as few processors as possible. A proper implementation expected to give a better speedup in dual core machine and in a quad core machine. In this paper, the implementation of different parallel algorithms such as Naive Algorithm , Brute Force Algorithm , Graham Scan Algorithm on geometric problems is presented.

Keywords: Parallel Algorithms, Dual Core and Quad Core Machines, Naive Algorithm , Graham Scan Algorithm , Brute Force Algorithm , Geometric Problems.

I. INTRODUCTION

A parallel algorithm is the just opposite of serial algorithm. Parallel algorithms involve sub-computations whose amount of work is not known in advance, and hence the work can only be distributed at execution time. Computational geometry is the study of algorithms which can be stated in terms of geometry [1].

Some purely geometrical problems arise out of the study of computational geometric algorithms, and such problems are also considered to be part of computational geometry [2]. Computational geometry focuses heavily on computational complexity since the algorithms are meant to be used on very large datasets containing tens or hundreds of millions of points.

For large data sets, the difference between O(n^2) and O(n log n) can be the difference between days and seconds of computation. Some fundamental problems of this type are Convex hull , Line segment intersection , Delaunay triangulation, Voronoi diagram, Closest pair of points , Euclidean shortest path, Polygon triangulation and Mesh generation.

II. PARALLEL NAIVE ALGORITHM

Input - > x[i] , y[i] as the x and y- coordinates of point i

```
z[i] = (x[i])^2 + (y[i])^2
for i=0 to n-2
    for j=i+1 to n
        for k=i+1 to
            if(j!=k)
                zn=(x[j]-x[i])*(y[k]-y[i])-(x[k]-x[i])*(y[j]-y[i]);
                if(flag=(zn1<0)
                xn=(y[j]-y[i])*(z[k]-z[i])-(y[k]-y[i])*(z[j]-z[i]);
                yn=(x[k]-x[i])*(z[j]-z[i])-(x[j]-x[i])*(z[k]-z[i]);
                for m=0 to n
                    flag=flag&&((x[m]-x[i])*xn+(y[m]-y[i])*yn+(z[m]-z[i])*zn<=0);
                if (flag)
                printplot(i,j,k);
            end if
        end for
    end for
end for
```

In two dimensions, one way to detect if point D lies in the circumcircle of A, B, C is to evaluate the required determinant.

When *A*, *B* and *C* are sorted in a counterclockwise order, this determinant is positive if and only if *D* lies inside the circumcircle.The approach followed to parallelize the algorithm involves parallelizing the outer for loop. We have to keep in mind that all the variables involved except the point sets should be private i.e. each processor should have its own copy of the variable(s),this helps in eliminating race conditions and achieving a good speedup.

III. PARALLEL GRAHAM SCAN ALGORITHM

Sort all points in S based on their position on the X axis using parallel quicksort
Designate point left as the leftmost point
Designate point right as the rightmost point
Remove left and right from S
While there are still points in S
remove Point from S
if Point is above the line from left to right
add Point to the end of array upper
else
add Point to the end of array lower

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 2, Issue 3, March 2015

Construction of the lower hull on one processor:

Add left to lower_hull
While lower is not empty
add lower[0] to the end of lower_hull
remove lower[0] from lower
while size(lower_hull >= 3 and the last 3 points lower_hull are not convex
remove the next to last element from *lower_hull*

Construction of the upper hull on another processor:

Add left to upper_hull
While upper is not empty
add upper[0] to the end of upper_hull
remove upper[0] from upper
while size(upper_hull >= 3 and the last 3 points upper_hull are not convex
remove the next to last element from *upper_hull*
Merge upper_hull and lower_hull to form hull
return hull

We will implement this algorithm for Convex Hull .Graham Scan, as it is called, works by picking the leftmost point p, i.e. the one with the minimum p.x , then scanning the rest of the points in counterclockwise order with respect to p. As this scanning is done, the points that should remain on the convex hull are kept, and the rest are discarded leaving only the points in the convex hull at the end.

IV. PARALLEL BRUTE FORCE ALGORITHM

for each p in P:
for each q in P:
dis=dist(p,q)
if p ≠ q and dis < minDist:
minDist = dist(p, q)
closestPair = (p, q)
return closestPair

The closest pair of points can be computed in $O(n^2)$ time by performing a brute-force search. To do that, one could compute the distances between all the $n(n - 1) / 2$ pairs of points, then pick the pair with the smallest distance. The parallel approach involves parallelizing the outer for loop. The variable "dis" has to be kept private since each processor calculates the distance independently and hence should keep a copy of the variable itself to avoid race conditions. The "minDist" variable should be kept shared as both the processors should update it.

V. PERFORMANCE ANALYSIS

SL. NO	NO. OF POINTS	TIME TAKEN(Serial)	TIME TAKEN(Parallel)	SPEED UP
1	300	4.307	2.970	1.449
2	350	7.790	5.401	1.442
3	400	13.846	8.806	1.572
4	450	20.785	13.775	1.508
5	500	31.387	21.711	1.445

Table 1: Performance Analysis Table of Parallel Naive Algorithm

SL.NO.	NO. OF POINS	TIME TAKEN(Serial)	TIME TAKEN(Parallel)	SPEED UP
1	200000	0.075	0.048	1.547
2	400000	0.158	0.103	1.538
3	600000	0.332	0.217	1.529
4	800000	0.452	0.286	1.579
5	1000000	0.605	0.388	1.556

Table 2: Performance Analysis Table of Parallel Graham Scan Algorithm

SL. NO	NO. OF POINTS	TIME TAKEN (Serial)	TIME TAKEN(Parallel)	SPEED UP
1	8000	0.469	0.314	1.493
2	12000	1.053	0.674	1.561
3	16000	1.863	1.173	1.587
4	20000	2.911	1.823	1.596
5	24000	4.192	2.631	1.593

Table 3: Performance Analysis Table of Parallel Brute Force Algorithm

VI. CONCLUSION

In this paper, we have discussed the implementation of different parallel algorithms on geometric problems and the performance tables related to different parallel algorithms are presented. A proper implementation helps in speed up the operation. This shows that parallel programming is better approach to solve any problem efficiently as compared with serial algorithms.

REFERENCES

[1] Dedu,Vialle and Timsit, "Comparison of OpenMP and classical multithreading parallelization for regular and irregular algorithms," In Fouchal, Software Engineering Applied to Networking Parallel/Distributed Computing (SNPD), Association for Computer and Information Science ,53–60,2000.
[2] Nikolopoulos, D.S.Polychronopoulos and C.D.Ayguade, "Scaling irregular parallel codes with minimal programming effort. In Supercomputing," Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (CDROM), ACM Press ,2001.
[3] S¨uß and Leopold, " Common mistakes in OpenMP and how to avoid them," In Proceedings of the International Workshop on OpenMP - IWOMP'06,2006
[4] Bull, J.M. and O'Neill, "A micro benchmark suite for OpenMP 2.0," SIGARCH Compute. Archit. News 29(5) ,41–48,2001
[5] F.P. Preparata and S. Hong, "Convex hulls of finite sets of points in two and three dimensions," Comm. ACM, Vol. 20,pp. 87-93,1977.
[6] H. Freeman and R. Shapira, "Determining the minimum-area encasing rectangle for an arbitrary closed curve," Comm. A.C.M., Vol. 18,pp. 409-413,July 1975.
[7] T. Lozano-Perez, "An algorithm for planning collision-free paths among polyhedral obstacles," Comm. ACM, Vol. 22, pp. 560-570,1979.
[8] L. O. Chua and L. Yang , "Cellular neural networks: Theory," IEEE Trans. Circuits Syst., vol. 35, pp.1257 -1272 ,1988.
[9] L. O. Chua and L. Yang, "Cellular neural networks: Applications," IEEE Trans. Circuits Syst., vol. 35, pp.1273 -1290,1988.
[10] T. Roska and L. O. Chua ,"The CNN universal machine Part 2: Programmability and applications," Proc. IEEE CNNA-92, pp.181 -190 1992 .
[11] L.J. Guibas and F.F. Yao, "On translating a set of rectangles," Proc. of the Twelfth Annual ACM Symposium on Theory of Computing, Los Angeles, pp. 154-160,April 1980.
[12] J. O'Rourke, "An on-line algorithm for fitting straight lines between data ranges," Comm. ACM, Vol. 24, pp. 574-578, September 1981.
[13] H. M. Alnuweiri and V. K. P. Kumar ,"Efficient image computations on VLSI architectures with reduced hardware," Proc. IEEE 1987 Workshop Comput. Architecture, Pattern Anal. Mach. Intell., pp.192 -199,1987.
[14] P. J. Burt and G. S. van der Wal, "Iconic image analysis with the pyramid vision machine (PVM)," Proc. IEEE 1987 Workshop Pattern Anal. Mach. Intell., pp.137 -144,1987.
[15] Î'. Chazelle, "Computational geometry on a systolic chip," IEEE Trans. Comput., vol. C-33, pp.774 -785 ,1984.
[16] P. Clermont and A. Merigot ,"Real time synchronization in a multi-SIMD massively parallel machine," Proc. IEEE 1987 Workshop Pattern Anal. Machine Intell., pp.131 -136,1987.
[17] R. Cole ,"Parallel merge sort," Proc. 27th IEEE Symp. Foundations Comput. Sci., pp.511 -517 ,1986 .