High Performance Computing over Geo-Spatial data: A summary of results

Satish Puri

Advisor: Dr. Sushil K. Prasad

Department of Computer Science, Georgia State University

I. Introduction

Polygon overlay is one of the complex operations in Geographic Information Systems (GIS). In GIS, a typical polygon tends to be large in size often consisting of thousands of vertices. Sequential algorithms for this problem are in abundance in literature and most of the parallel algorithms concentrate on parallelizing edge intersection phase only. Moreover, spatial data files tend to be large in size (in GBs) and the underlying overlay computation is highly irregular and compute intensive. Motivated by MapReduce programming model and GPGPU, we propose to develop and implement scalable distributed algorithms to solve large-scale overlay processing.

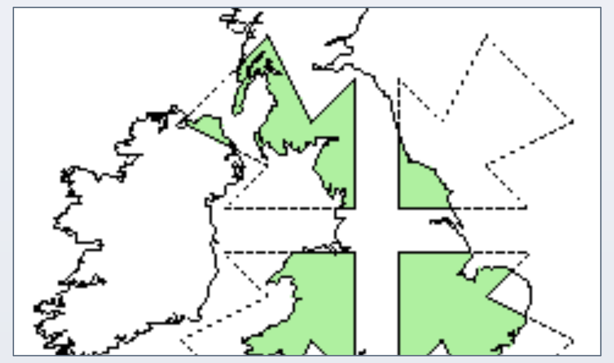


Fig. 1 Intersection of two polygonal maps

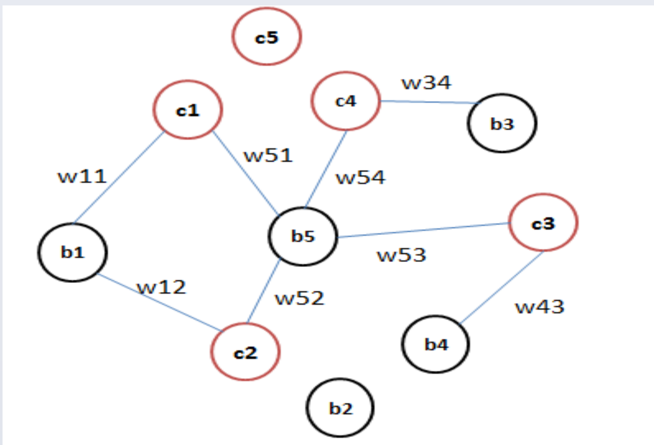
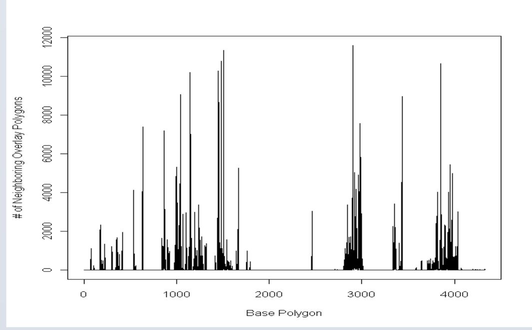


Fig. 2 Overlap Graph for polygons from two map layers



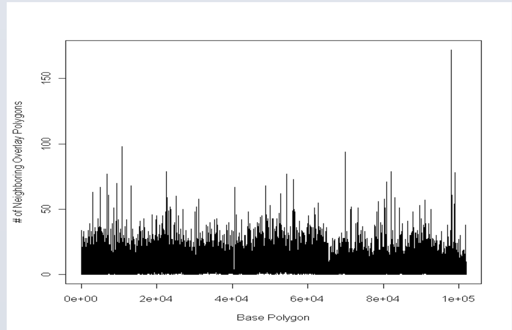


Fig. 3 and 4 shows load distribution for two sets of polygonal data

II. Related work

1. Ealier approaches include parallelization of Sutherland-Hodgman polygon clipping algorithm and Liang-Barsky line clipping algorithm.
2. MapReduce implementation focuses on spatial join and distributed R-tree implementation.

III. Problem description

Polygon overlay combines the input polygons from different maps into a single new map. The input to binary map overlay are two map layers $L_1 = [p_1, p_2, .., p_n]$ and $L_2 = [q_1, q_2, .., q_m]$ where p_i and q_i are polygons represented as (x, y) co-ordinates of vertices. The output of the overlay operation is a third layer $L_3 = L_1 \times L_2 = [o_1, o_2, .., o_k]$ represented by k output polygons and this output depends on the overlay operator denoted as ×, × ∈ {Union, Intersection, XOR, Difference} determines how map layers are combined.

IV. Technical Contributions

1. O(log n) time polygon clipping algorithm using plane sweep method using O(n+k+k') and O(log n) time algorithm using graph traversal method with O(n+ k) number of processors using plane-sweep tree in PRAM model [1,3].
2. Design and implementation of an end-to-end spatial overlay system using Hadoop MapReduce platform and MPI.
3. Absolute speedup of 44x using R-tree indexing and grid partitioning using MPI compared to ArcGIS 10.1 and relative speedup of 22x using Hadoop MapReduce platform [1,2]

Algorithm 1 Algorithm Parallel Polygon Overlay

INPUT: $V = V_b \cup V_o$ and $E = E_b \cup E_o$

Step 1: Sort the vertices in V by their y-coordinates.

Step 2: Partition E into k subsets $E_1, E_2, \ldots, E_k$ of edges where E_i belongs to ith scanbeam.

Step 3:

for all E_i in E **in parallel do**

  Get y-coordinate of lower(y_b) and upper(y_t) scanline for E_i

  Step 3.1: $P_i \leftarrow$ ProcessIntersection(y_b, y_t)

  Step 3.2: $P_i \leftarrow P_i \cup$ ProcessEdgeEndPoint(y_b, y_t)

end for

Step 4: $P \leftarrow (P_1 \cup P_2 \cup \ldots \cup P_n)$

Fig. 5 Here, input is a pair of overlapping polygons. The polygon edges are divided into scanbeams and each scanbeam is processed in parallel. Output is zero or more polygon(s).

V. Different Architectures for Distributed Overlay System

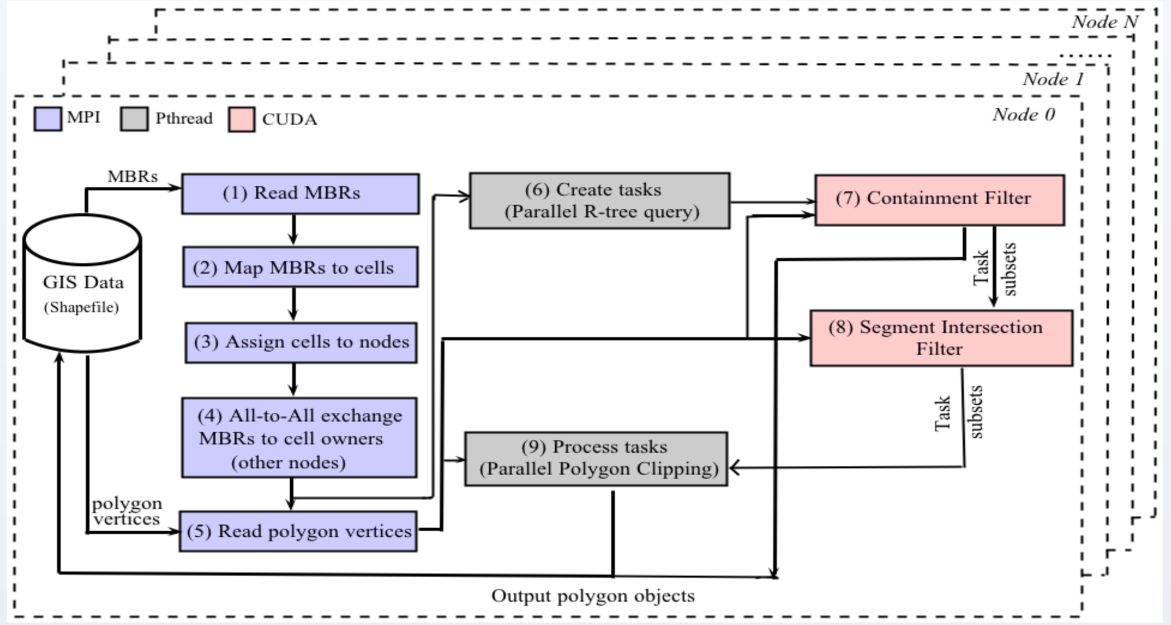


Fig. 6 MPI-GIS with Pthread and CUDA based overlay

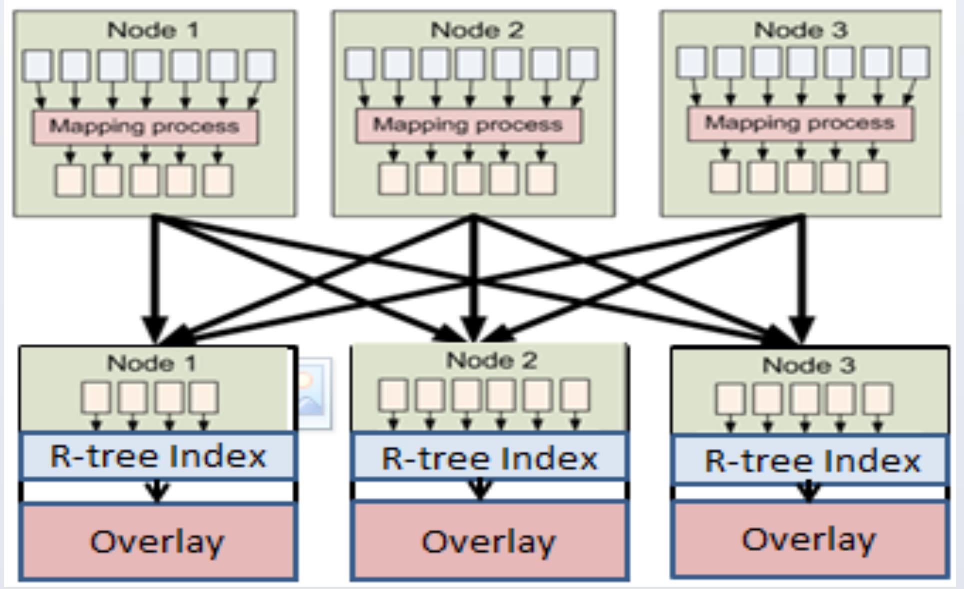


Fig. 7 Two level of spatial indexing. Grid based partitioning and R-tree based indexing. Polygons are mapped based on grid cells. Each reducer performs overlay processing for one grid cell.

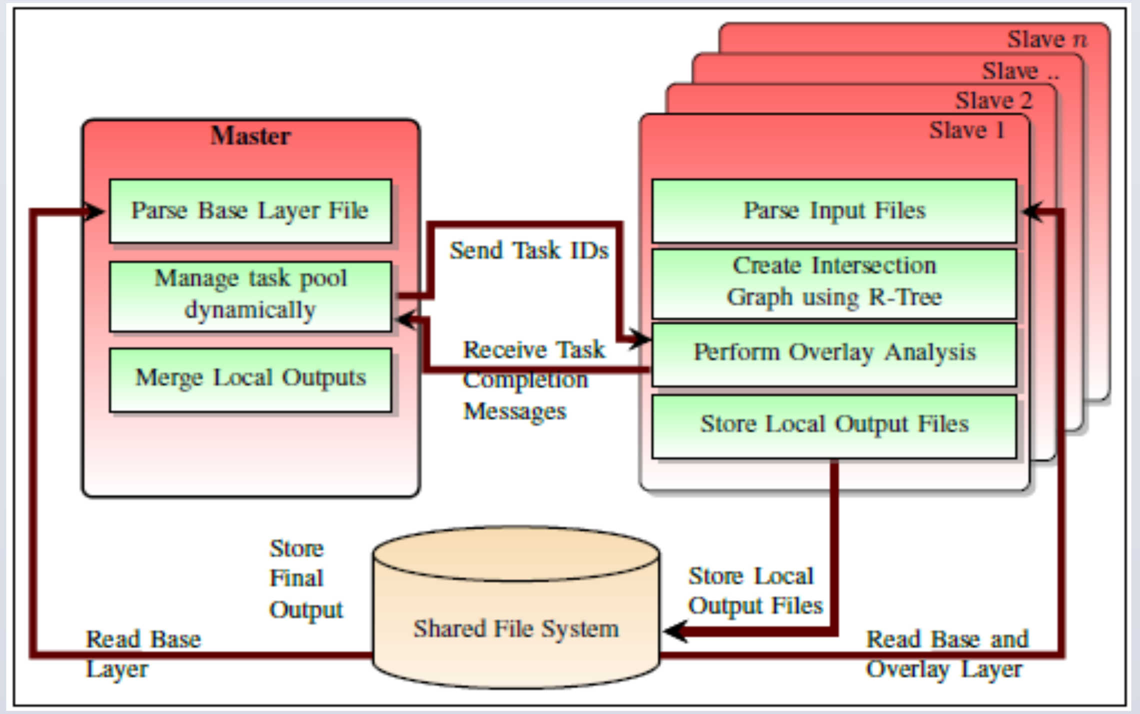


Fig. 8 Our MPI based Overlay system based on dynamic load-balancing and R-tree.

VI. Experimental Results

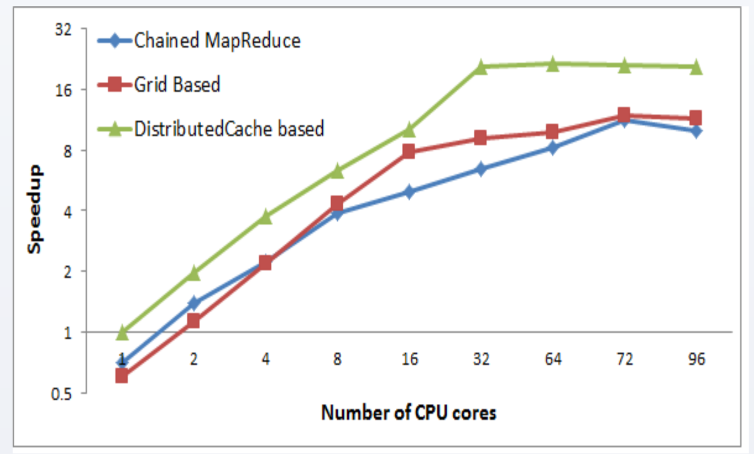


Fig. 9 Speedup of the three versions using data set I having 4K * 1M polygons for three versions a) Chaining of two MapReduce jobs b) Grid-Rtree based with map and reduce phase c) Distributed Cache based with a single map phase only

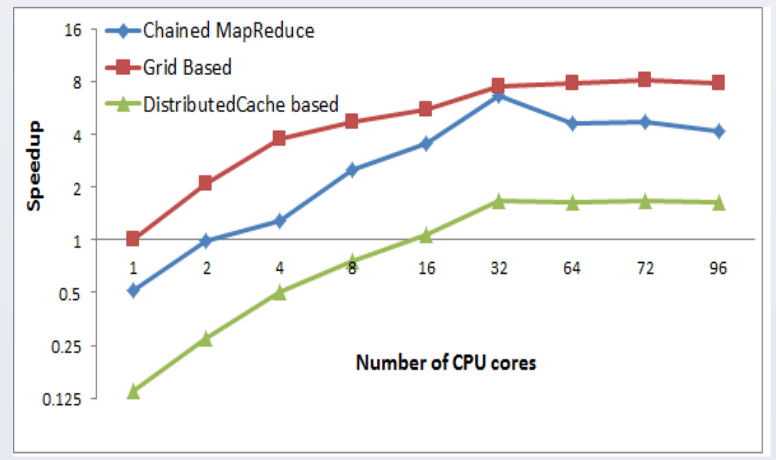


Fig. 10 Speedup of the three versions using data set II having 300K * 750K polygons for three versions a) Chaining of two MapReduce jobs b) Grid-Rtree based with map and reduce phase c) Distributed Cache based with a single map phase only

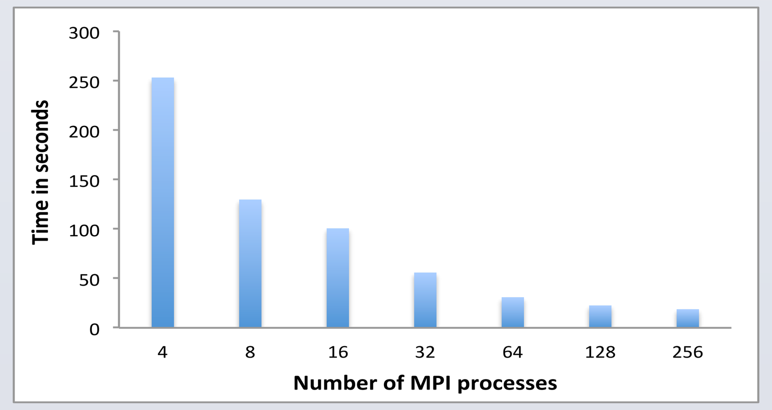


Fig. 11 Scalability of MPI-GIS using USA shapefile data set

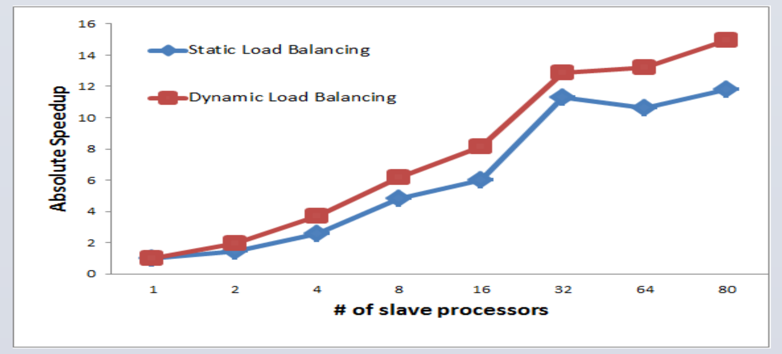


Fig. 12 Absolute speedup of 15x for MPI based overlay system using 8K * 400K (data set III) polygons using 80 CPU cores

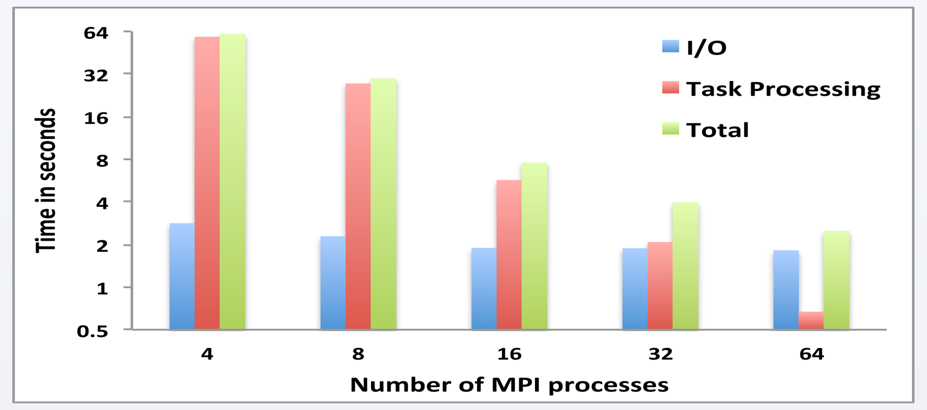


Fig. 13 Breakdown of execution time for MPI-GIS using GML data set

VII. Future Work

1. GPU implementation of Segment tree and TPR tree.
2. Multi-layer overlay algorithm
3. Load-balanced overlay processing using MapReduce based CPU-GPU cluster.

VIII. References

1. S. Puri and S.K. Prasad, *Output-Sensitive Parallel Algorithm for polygon clipping*, International Conference on Parallel Processing (ICPP), 2014.
2. S. Puri, D. Agarwal, X. He, and S. K. Prasad, *MapReduce algorithms for GIS polygonal overlay processing*, IEEE International Parallel and Distributed Processing Symposium Workshops , 2013.
3. S.Puri, S.K. Prasad, *Efficient Parallel and Distributed Algorithms for GIS Polygonal Overlay Processing*, in IEEE IPDPS PhD Forum, 2013
4. S. Puri and S.K. Prasad, MPI-*GIS: New Parallel Algorithm and System Prototype*, Technical Report, DiMoS Lab. GSU, 2014
5. D. Agarwal, S. Puri, X. He, and S. K. Prasad, *A system for GIS polygonal overlay computation on linux cluster - an experience and performance report*, in IEEE International Parallel and Distributed Processing Symposium workshops, 2012.
6. D. Agarwal, S. Puri, X. He, and S.K. Prasad, *Cloud Computing for Fundamental Spatial Operations on Polygonal GIS Data*, in: Cloud Futures 2012 - Hot Topics in research and education, Berkeley, California, 2012.
7. B. Vatti, *A generic solution to polygon clipping*, Communications of the ACM, vol. 35, no. 7, pp. 56-63, 1992.