A Heterogeneous HPC Platform for Ill-Structured Spatial Join Processing

Danial Aghajarian (daghajarian@cs.gsu.edu), Advisor: Sushil K. Prasad (sprasad@gsu.edu)

Computer Science Department, Georgia State University

Introduction

- Real spatial objects such as polygons, e.g. representing places on Earth, can have up to hundreds of thousands edges.
- Spatial applications such as Geographical Information Systems (GIS) are interested in processing of very large sets of these objects (Millions).
  - Urban Planning and Management: Where is the best place to put a hospital that it is easily accessible to as many people as possible?
  - Electronics: What is the best routing configuration for a given electronic circuit?
- Variations of spatial join are the most popular spatial processing with O^2 time.
- We propose a distributed heterogeneous HPC platform for spatial join processing based on two-step filter and refinement approach.
- First, we introduce a set of GPU-suited data structures and algorithms.
- Second, several GPU-based spatial join systems to improve the performance of current state of the art systems.

Contribution Summary

- Part 1: Algorithms
  - SMF: a Sort-based MBR overlap Filter (**A.1**)
  - CMF: a filter based on Common MBR (**A.2**)
  - CMF-grid: grid version of CMF (**A.3**)
- Part 2: Spatial Join Systems
  - GCMF: GPU-based spatial join system for ST_intersect predicate (**S.1**)
  - GCMF+: Improved version of GCMF based on CMF-grid (**S.2**)
  - MPI-Cuda-GIS: A distributed MPI-based spatial join system over heterogeneous cluster (**S.3**)

Problem Definition

Input: Two sets (layers) of polygons.

Output: Cross-layer pairs of polygons that share any portion of space.

Notations:

MBR_P1: Minimum Bounding Rectangle (MBR) of polygon P1. (Figure 4-Blue and black MBRs)

MBR_P1^P2: Common MBR of MBR_P1 and MBR_P2. (Figure 4-Green MBRs)

E_P: List of edges of polygon P. **E_P(i)**: ith edge of P.

A.1: SMF: an MBR overlap test algorithm

Input: Minimum Bounding Rectangles (MBR) of polygonal objects.

Output: Set of MBR overlapping objects (**Set C**).

Algorithm:

1. Build data structures: Sorting MBRs over one dimension.
   - ❑ sortIndex **(d):** Sort index of MBR coordinate in one dimension.
   - ❑ rankIndex **(e):** Rank of each MBR in sorted array.
2. For each MBR, take the MBRs lie in its sorted interval range and if they overlap in another dimension too, add the pair to the output

Results Summary:

- ✓ Running time of SMF filter is linear.
- ✓ Space complexity is output sensitive.
- ✓ 57-fold speedup versus *GEOS optimized library.*

A.2: CMF: Common MBR filter

Input: Set C and their corresponding objects.

Output: Set W, I.

CMF classifies **set C** into three following categories:

1. ***Within candidate set (W):*** set of all the polygon pairs (P1,P2) in **set C** such that MBR_P1^P2 is either equal to MBR_P1 or MBR_P2. **(**Figure 4-a)
2. ***Intersecting-edge candidate set (I):*** set of all polygon pairs (P1,P2) in C such that (P1,P2) is not in W and sets E_P1 and E_P2 are non-empty. **(**Figure 4-b)
3. ***Disjoint set:*** Polygon pairs (P1,P2) in **set C** that are neither in B nor in **I**. **(**Figure 4-c)

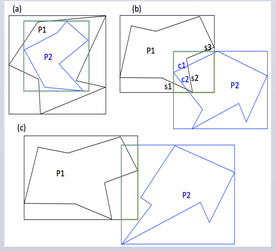

Figure : CMF classifies pairs in set C into three subsets.

Results Summary:

- ✓ CMF reduces %65 of **SMF output** pairs before refinement phase.
- ✓ CMF makes edge-intersection test 30 times faster
- ✓ CMF eliminates workload by a factor of 40.
- ✓ It always reduces workload of point-in-polygon test independent of dataset.

A.3: CMF-grid: Common MBR grid filter

Unique properties of CMF-grid technique:

1. *grid-cells do not necessarily cover the whole universe.*
2. *grid-cells may overlap with each other (in cases that Common MBRs of overlapping pairs overlap).*

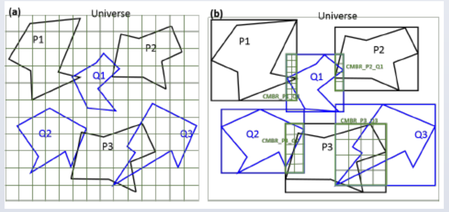

Figure : (a) uniform grid technique, and (b) CMF-grid. In CMF-grid, cells are not of the same size and may not cover the whole universe.

Results Summary:

- ✓ CMF-grid reduces workload of the refinement phase by factor of **30,000** compared to all-to-all
- ✓ In order to achieve the fastest end-to-end running time, CMF-grid does not need to minimize the workload by applying a very fine grid; 4 to 6 times coarser grid-cell sizes can lead to best performance.

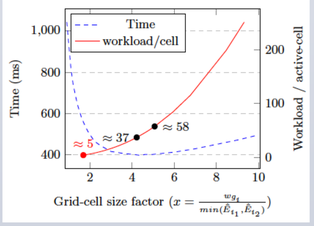

Figure : The impact of grid size (w_g_i) on total running time of CMF-grid and edge-intersection test. The minimum time happens in the range 4 to 6.

CMF vs CMF-grid

	All-to-all	*CMF* [11]	***CMF-grid***
Time (ms)	120,751	981	**89**
Pairs $\in I$	566,656	198,142	198,125
Edges(all)	2,002,910,863	46,421,188	**147,002,513**
Edges(active)	2,002,910,863	46,421,188	**15,043,183**
Workload	535, 108, 085, 968	12,794,606,592	**17,370,352**

Results Summary:

- ✓ Refinement phase after CMF is more than **50%** of total time
- ✓ Refinement phase after CMF-grid is less than **15%** of total time
- ✓ CMF-grid reduces workload of the refinement phase **700 times** more than CMF.

S.1: GCMF System

A GPU-based spatial join system based on ST_intersect

1) ***SMF***: A sort-based MBR filtering algorithm (**A.1**).
2) **CMF**: A filter based on CMBR (**A.2**).
3) **PnP-Test**: A load-balanced implementation of parallel *point-in-polygon* test.
4) **EI-Test:** A load-balanced edge-intersection parallel implementation based on all-to-all test in CMBR.

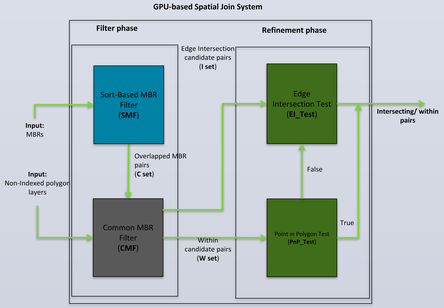

Figure : GCMF system workflow.

S.2: GCMF+ System

A upgraded GCMF with CMF-grid components

1) Filter phase is the same as GCMF.
2) **EI-Test:** A load-balanced edge-intersection parallel implementation based on in-cell all-to-all test.

GCMF vs GCMF+

Dataset	Running Time (ms)			
	Sequential		Parallel	
	PostGIS	GEOS	GCMF [11]	**CMF-grid**
Urban	3,120	5,770	52	31
Water	232,122	148,040	1,663	739

Speedup Results:

- ✓ GCMF+ achieves up to 200-fold speedup versus GEOS optimized library.
- ✓ GCMF+ achieves 2.225 fold speedup versus GCMF.

S.3: MPI-Cuda-GIS System

A Big Spatial Data Processing System over Distributed GPU-equipped Clusters

Goal: Handling TB-size spatial data for spatial join processing

Challenges:

1) **Load balancing and task partitioning:** Minimizing data overhead and distributing data as evenly as possible.
2) **Scaling**: Holding weak scalability condition.
3) **IO**: MPI-IO, distributed databases such as Cassandra

RESEARCH POSTER PRESENTATION DESIGN © 2015
www.PosterPresentations.com