A Heterogeneous HPC Platform for Ill-Structured Spatial Join Processing

Doctoral ShowCase at SC2017 Presentation by

Danial Aghajarian
Advisor: Sushil K. Prasad

Computer Science Department
Georgia State University

Fall 2017

Outline

- Introduction
- Motivation
- Research and Development Agenda – Part I
  - Algorithms and Data Structures
- Research and Development Agenda – Part II
  - Spatial Join Systems
- Conclusion
- Timeline and Publications

Introduction

- Urban planning and management:
  - "Where is the best place to put a hospital that it is easily accessible to as many people as possible?"
  - "Which houses are within the 100-year flood level of a river?"
- Politics, micro-economics:
  - "How is the distribution of wealth in different areas?"
- Electronics:
  - "What is the best routing configuration for a given electronic circuit?"
- Transportation management, Astronomy

A Geographic Information System (GIS) is a great tool to answer these types of questions through **Spatial Analysis**

Spatial Analysis: various mathematic and geometric analysis over spatial data to extract useful information.

Example: Spatial Join Processing

Irrelevant Fields?!, Irrelevant Questions?!

Introduction

- Spatial join:
  - Input: two layers of spatial data and a **predicate**
  - Output: pairs of cross-layer objects satisfying the predicate
- Spatial data:
  - Raster data: matrix of cells (pixels)
    - Examples: satellite images, pictures, scanned maps
    - Pros: continuous surfaces, Cons: large in size, spatial inaccuracy
  - Vector data: set of lines and connected vertices
    - Examples: polygons, polylines, points
    - Pros: smaller size, Cons: silvers, overshoots, undershoots
- **Predicate**: KNN, point-in-polygon, overlay, ST-intersect, etc.
  - ✧Example: ST-intersect: If objects share any portion of space.

Datasets – Vector data

- Urban:
  - admin_states
  - urban_areas
- Telecom:
  - GA_telecom_base
  - GA_telecome_overlay
- Water:
  - US block boundaries: Census demographic information in US
  - US water bodies

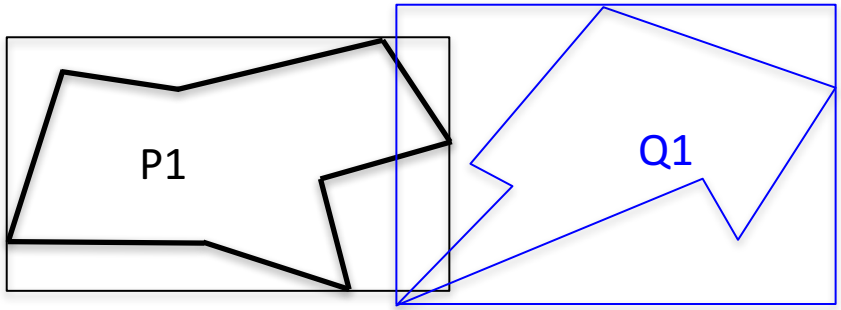
Label	Dataset	Polygons	Size
Urban	*ne_10m_admin_states*	11,878	$46MB$
	ne_10m_urban_areas	4,646	$41MB$
Telecom	*GA_telecom_base*	101,860	$171MB$
	GA_telecom_overlay	128,683	$240MB$
Water	*US_block_boundaries*	219,831	$2.175GB$
	US_water_bodies	463,591	$921MB$

Introduction

- Spatial Join is a computationally-intensive
- Two-steps spatial join technique:
  1. Filter phase: Reduce all possible cross-layer polygon pairs to a set of potentially satisfying pairs.
     - Most popular algorithm: MBR overlap (**O(1)**).
  2. Refinement phase: Removes all the unsatisfying pairs.
     - It usually involves local or global all-to-all tests (**O(n^2)**).
- [Wang2014] showed that **refinement phase** takes more than **5 times the filtering**

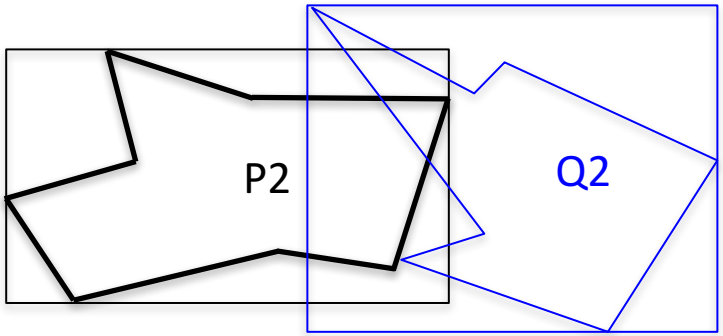
Introduction

An example of two-steps technique:



Layer 1: {P1, P2}

Layer 2: {Q1, Q2}



All possible pairs: (P1, Q1), (P1, Q2), (P2, Q1), (P2, Q2)

1) Filter: (P1, Q1), (P2, Q2)

2) Refinement: (P2, Q2)

Motivation

- Spatial Join filtering is an ill-structured problem:
  - Distribution of data may impact on efficiency of algorithms.
  - There are many alternative solutions.
  - Output is not always the same for different algorithms
  - Shape of objects may impact on efficiency of algorithms
- Unexplored efficient filtering techniques
- Lack of efficient parallel algorithms.
- Lack of efficient GPU-based spatial join systems

Research and Development Agenda

1. Algorithms and Data structures:
   - Sort-based MBR Filter (SMF):
     - A sort-based MBR overlap test algorithm
   - Common MBR Filter (CMF):
     - Linear time filter
     - Based on Common MBR (CMBR) area of two MBRs
   - CMF-grid:
     - Applying a non-uniform grid technique over CMBR

Research and Development Agenda

2. GPU-based spatial join processing systems:
   - **GCMF**:
     - a GPU-based end-to-end spatial join system
     - ST_intersect predicate
     - Non-indexed datasets
   - **GCMF+**:
     - Same as GCMF but with grid-CMF filter
   - Polygon overlay system
   - **MPI-CUDA-GIS**:
     - Distributed heterogeneous spatial processing system

Part I

Algorithms and Data Structures

1) Sort-based MBR Filter (SMF)

MBR Filter– Literature Review

1. Plane-sweep algorithms
   - SSSJ:
     - Sorting based on lower coordinate
   - Tree-sweep:
     - Using interval tree
   - List-sweep:
     - Using linked list for MBRs
   - Forward-sweep (PBSM):
     - Using linked list for MBRs
     - Forward scan
   - Strip-sweep:
     - Partitioning the domain into s trips and using List-sweep for each

- Sorting intervals
- Inherently sequential algorithms
- Link list data structure as the base data structure

MBR Filter– Literature Review

2. Tree-based algorithms
   - R-tree
   - Quad-tree/Oct-tree:
   - B+-tree
   - X-tree

- Hierarchical data structure
- Inefficient to parallelize
- Require recursions for querying

SMF – Algorithm properties

- GPU-based algorithm for MBR overlap test
- It is not a plane-sweep algorithm: Sorting coordinates instead of intervals.
- Linear space complexity.
- Time complexity:
  - n, m: number of MBRs in each layer
  - w: average MBR width, W_a: Universe width

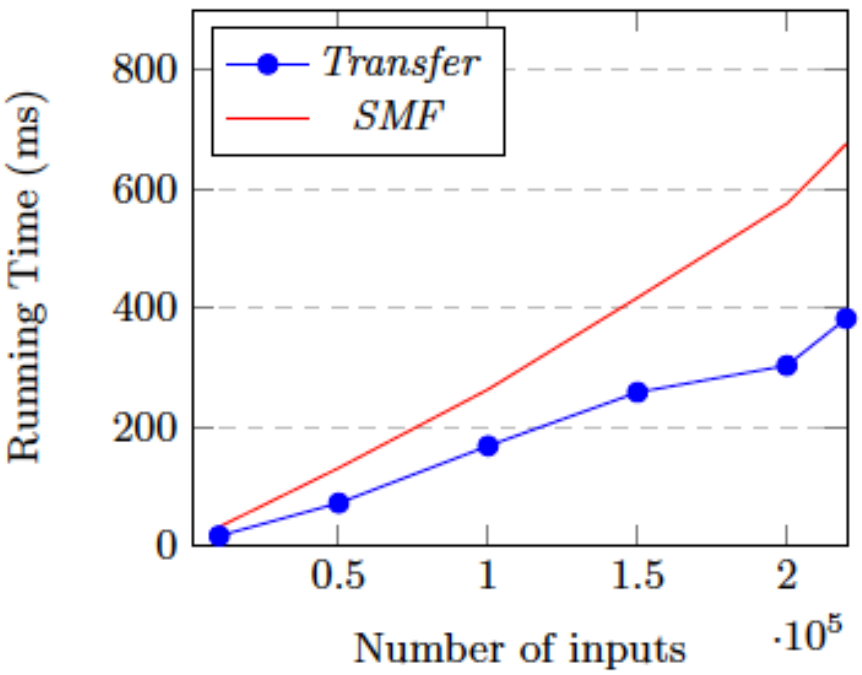
$$O((n+m)\cdot b + \frac{\bar{w}}{W_a}\cdot(n+m)^2)$$

- No duplicate output pairs. (**Lemma1**)
- No false positive. (**Lemma1**)
- Cons: output pairs are not sorted!

SMF – Performance

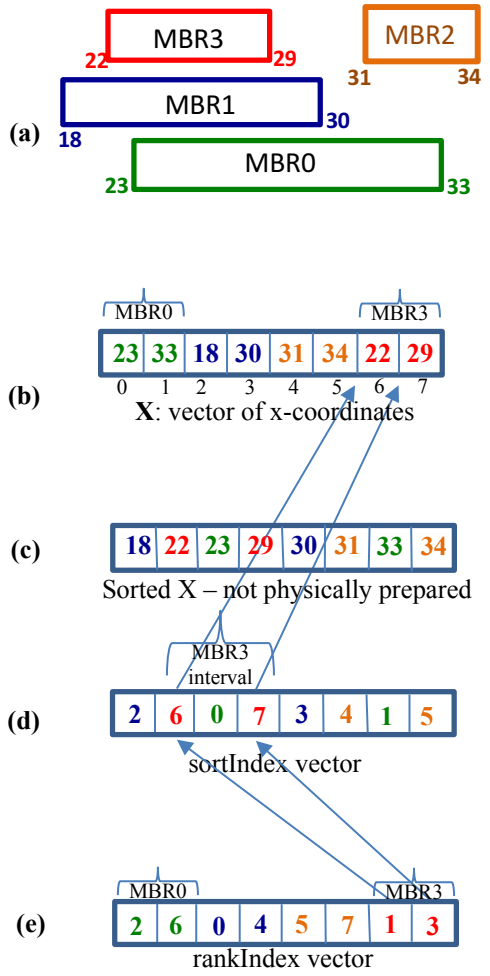
- **60-fold** speedup versus optimized GEOS library.
  - More than **220,000** by **460,000 MBRs**.
  - **13.05 Sec** VS **0.228 Sec**

- Scalability:
  - Fastest r-tree (Our lab 2015)
    - Speedup: **76-153** vs. sequential implementation.
    - Space: O(n^2).
    - **50,000** VS **1 Million** **MBR** in each layer.
  - Revised version (IPDPS 2017 – PhD forum): scalable but slower than SMF.



SMF - Data structure

- ✓ (a): An example of 4 MBRs.
- ✓ (b): input vector of x-coordinates.
- ✓ (c): Sorted x-coordinates.
- ✓ (d): sort indices of x-values.
- ✓ (e): rank Indices of x-values.



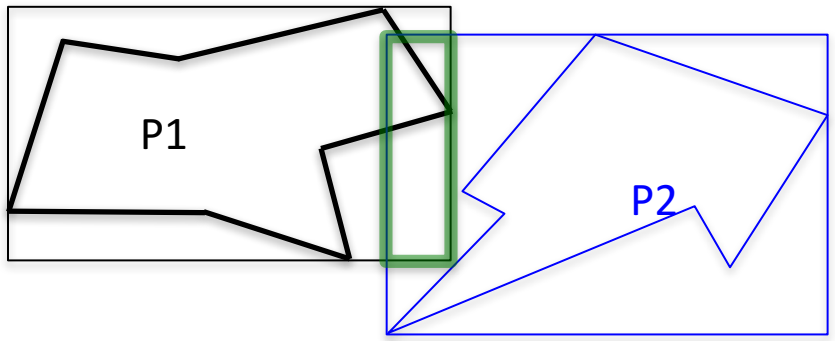
Part I

Algorithms and Data Structures

2) Common MBR filter (CMF)

Common MBR Filter (CMF)

- MBR overlap test: O(1)
- Refinement phase: O(n^2)
- Any other linear ideas for more filtering?



Common MBR (CMBR)

MBR resulting from the intersection of two overlapping MBRs

CMF

- CMF: edge-CMBR intersection test
  - O(n) , n=number of edges
  - Testing edges are independent from each other

1. Identifying more disjoint object pairs before refinement phase.

2. Eliminating non-intersecting edges from remaining candidate pairs.

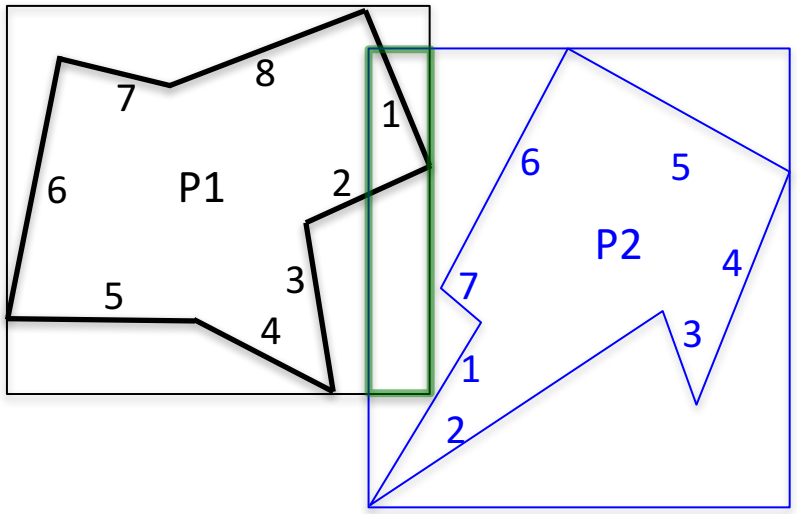
3. Reducing point-in-polygon work load.

CMF (Cont.)

1. CMF identifies more disjoint pairs before refinement phase:

❑ Considering just those edges that intersect with Common MBR for refinement phase (**Lemma3**).

✓ CMF filters out 65% candidate pairs before refinement phase over real datasets (**Disjoint pairs**).



- Without CMF:

  EI-test=|P1|*|P2|=8*7

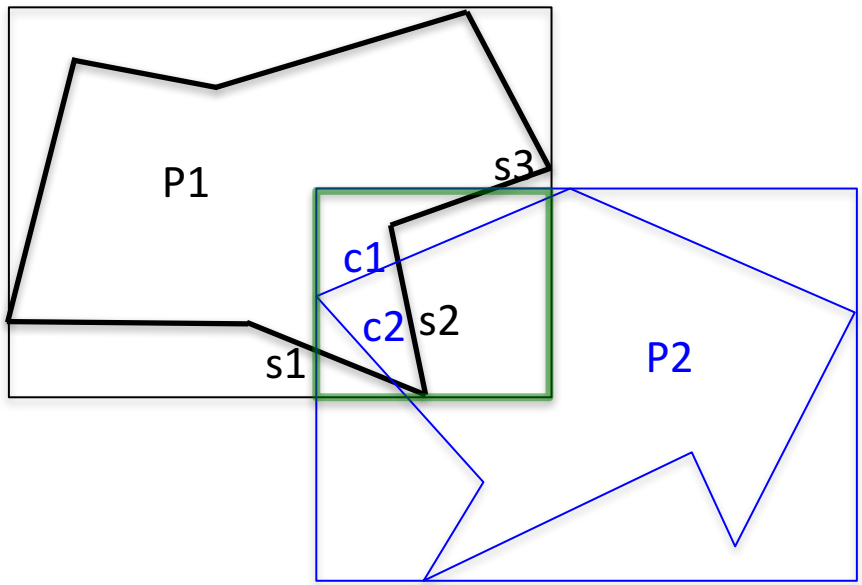
- With CMF:

  EI-test=|P1|*|P2|=0

CMF (Cont.)

2. CMF eliminates non-intersecting edges from remaining candidate pairs.
   - **Set I**: Pairs whose polygons intersect with CMBR

   – CMF reduces the size of polygons by factor of 40 for EI test.

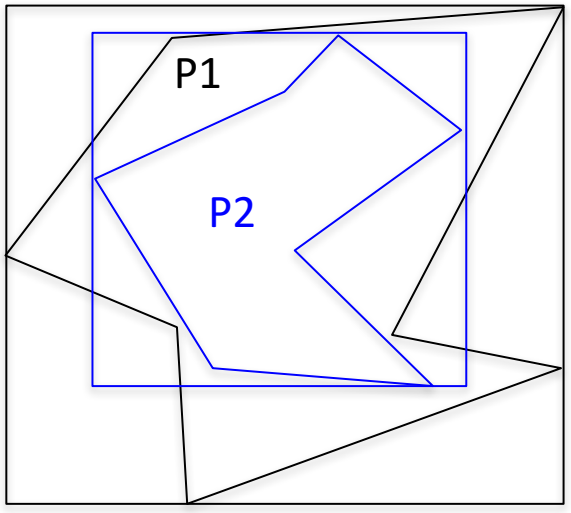


CMF (Cont.)

3. CMF reduces the number of PnP-test candidates.

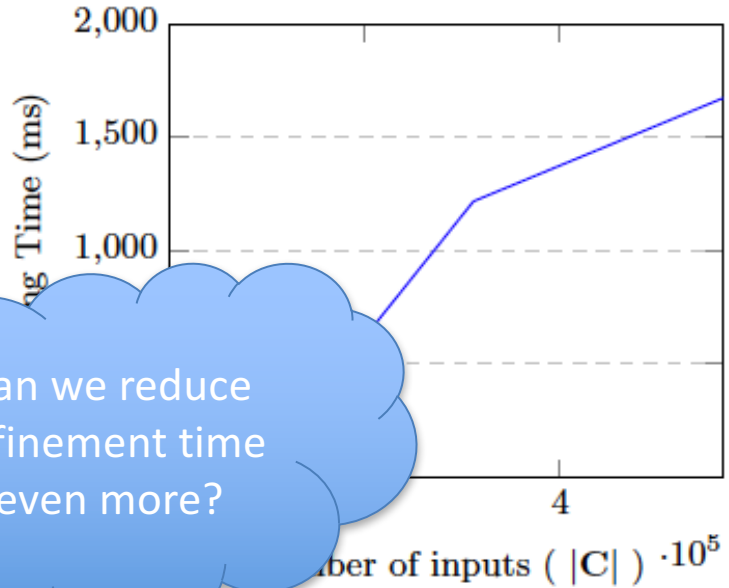
- **Set W:** CMBR=MBR(P1) or CMBR=MBR(P2) (**Lemma2**)
  - IF P1 is inside P2 THEN MBR(P1) is inside MBR(p2)

- For any given pair, CMF reduces the PnP test for either P1 inside P2 or P2 inside P1 (**Lemma2**)



CMF - Performance

- CMF running time is Linear.



More than 120 speedup for refinement pha

Can we reduce refinement time even more?

CMF effect on reducing the workload of the r
phase (Water dataset)

	No CMF	With MF
Time (ms)	120,751	**981**
# of edge-intersecting pairs	566,656	**198,142**
# of edges (layer1)	1,048,479,573	**25,969,322**
# of edges (layer2)	954,431,290	**20,451,866**

Part I

Algorithms and Data Structures

3) CMF-grid filter

CMF-grid

- Grid techniques in the literature:
  - Uniform grid methods
    - Universe is divided into same size cells
    - Poor choice for heterogeneous distribution of spatial objects in universe
    - Grid size: Too small vs too large
  - Non-uniform methods
    - Universe is divided into local regions
    - Then each region divides into more cells accordingly.
    - Quad-tree, Oct-tree

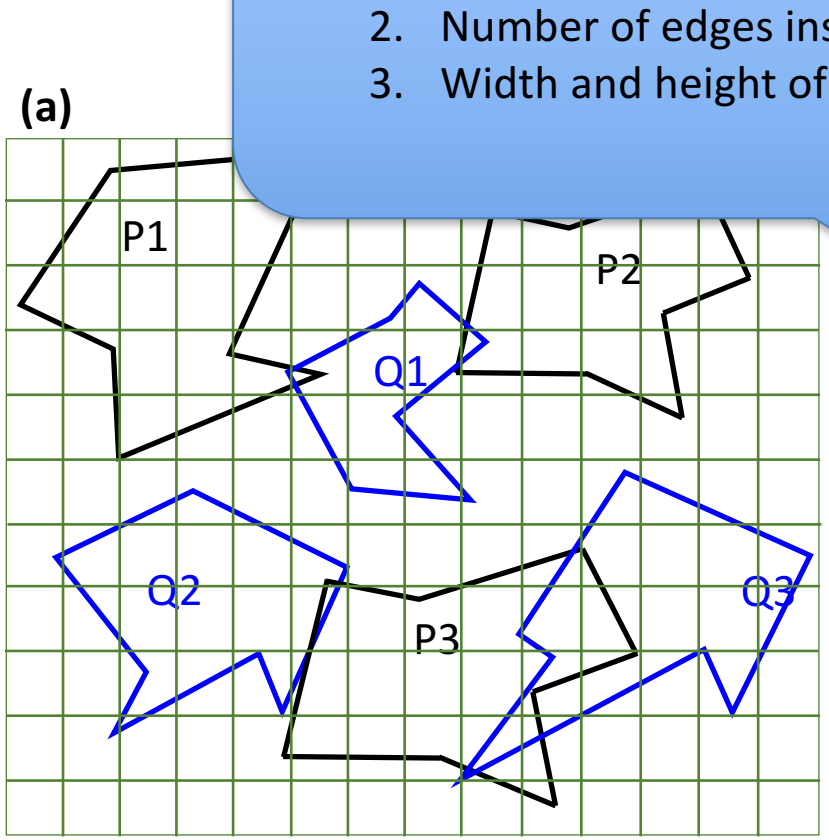
CMF-grid

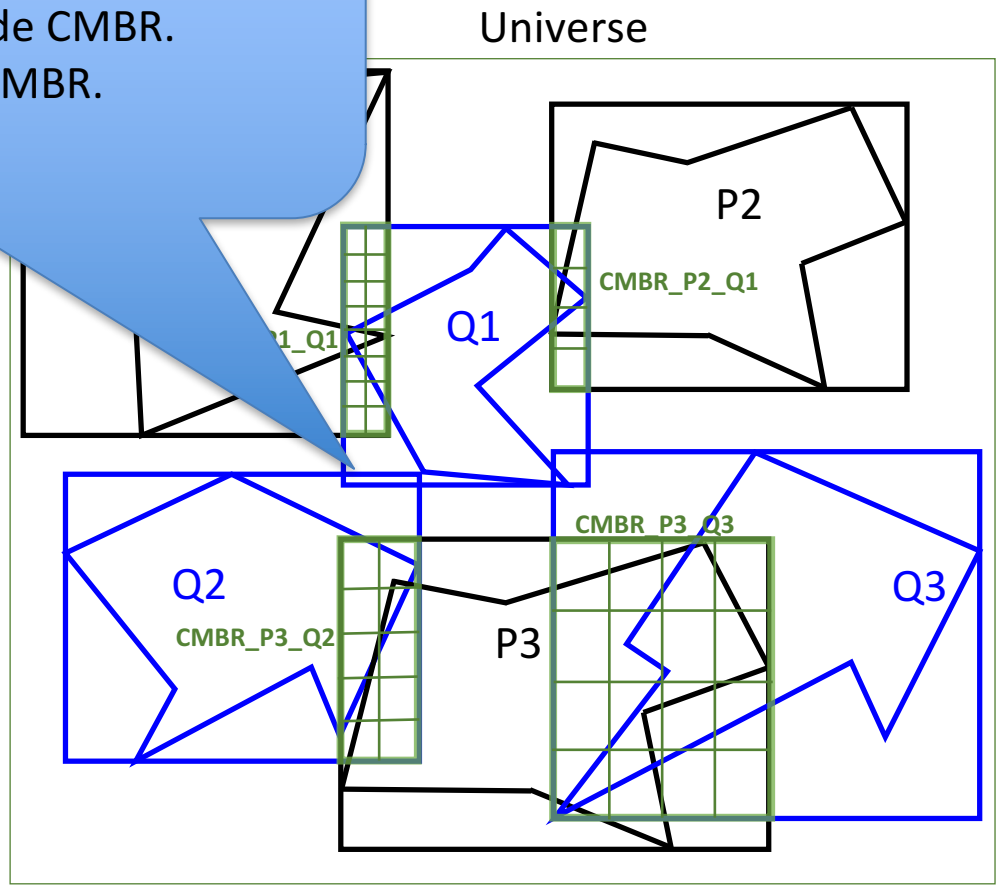
- Proposed technique: a non-uniform grid technique over Common MBR
  - Grid-cells may not cover the whole universe
  - Grid-cells may overlap with each other
  - Different cell sizes in various CMBRs
  - Same cell sizes within a CMBR

CMF-grid

- Size of cells in each CMBR is a function of:
  1. Average length of edges of that pair.
  2. Number of edges inside CMBR.
  3. Width and height of CMBR.

Regul **rid technique**





CMF-grid

- Challenges:

  ✓CMF-grid algorithm, implementation and proofs

  ✓Workload analysis based on average case scenario

  ✓Grid-cell property analysis:

  1. Cell shape:
     - Square
     - Rectangular (width-to-height ratio ?)
  2. Cell size

Algorithms and Data Structures

- Future directions:
  - Distributed algorithms
    - Multi-GPU single node
    - Multi-GPU multi node
  - New operations:
    - KNN, polygon overlay

Part II

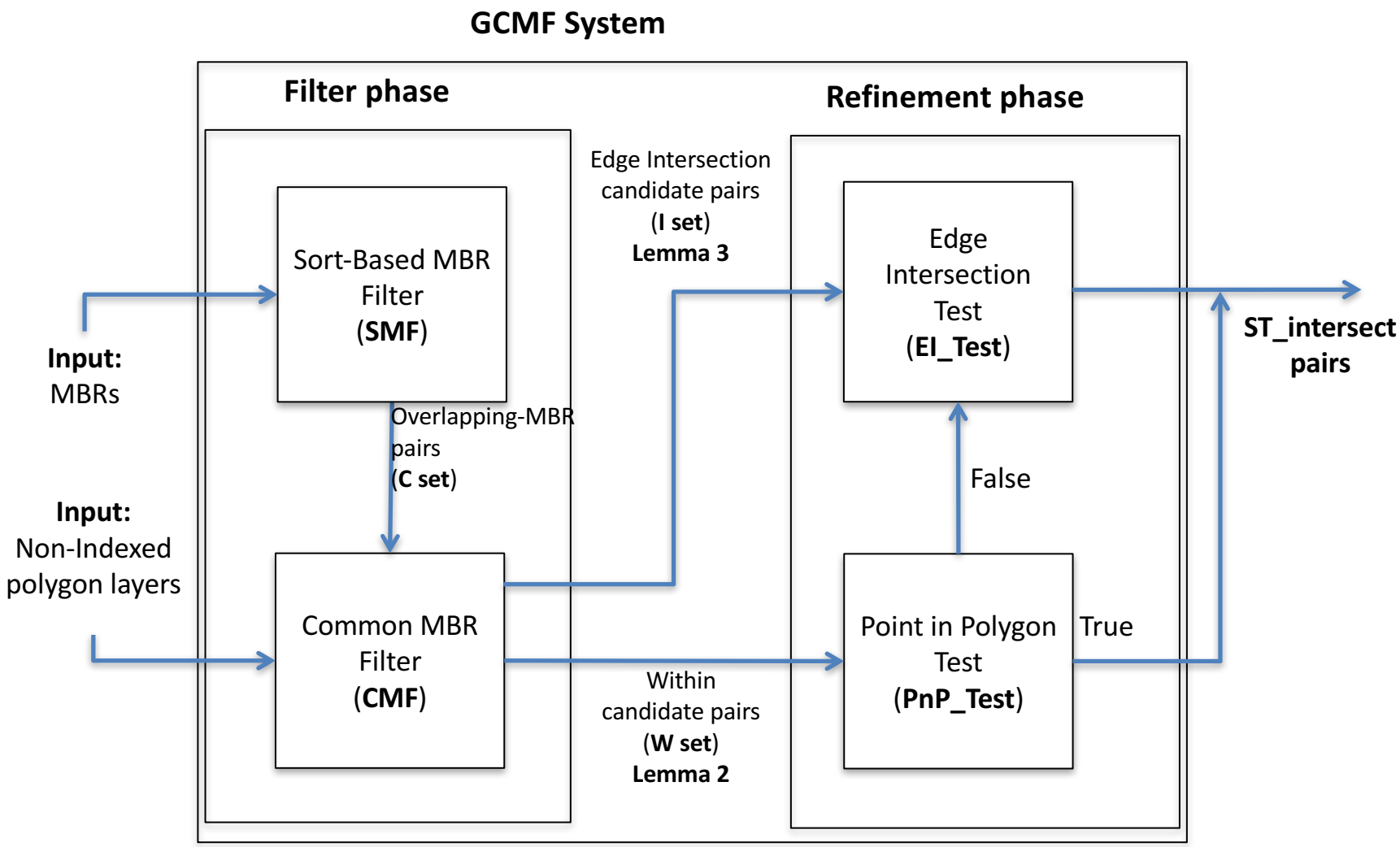
Spatial Join Processing Systems

Proposed system 1: GCMF

- GCMF: GPU-based system for ST_intersect predicate.
  - Detecting pairs whose objects share any portion of space.
    1. One polygon lies inside the other.
    2. They have at least one edge intersections

- Four components:
  - MBR filter test: our proposed Sort-based MBR filter (SMF)
  - CMF: Common MBR Filter
  - PnP: Point-in-Polygon test
  - EI: Edge-intersection test

System Overview



GCMF – PnP Test

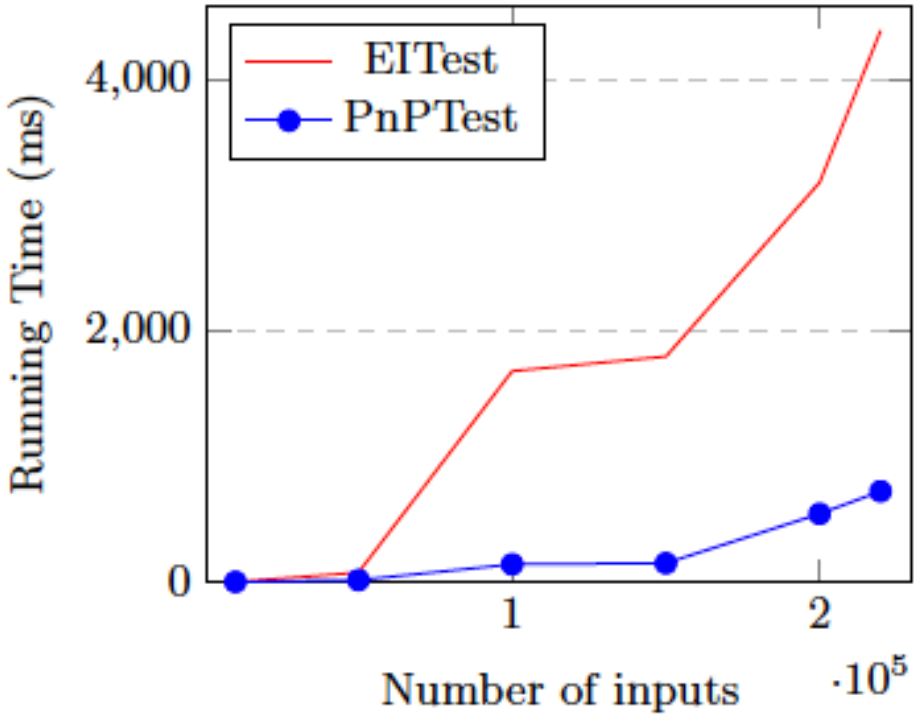
- **Best GPU work in literature**: CudaGIS [zhang2012CudaGIS]
  - Load balanced but coarse parallelism
  - Bottleneck: Large polygons
- Ray casting test over GPU
  - Designed for very large polygons with fine parallelism.
  - Load balanced within a GPU-block.

Running time of PnPTest versus sequential and naïve GPU implementation

Dataset	Running Time (ms)			Speedup	
	Sequential	GPU		PnP vs GEOS	PnP vs naïve GPU
	GEOS	Naive	PnPTest		
Urban	672	210	21	32	10
Telecom	165	45	6	27.5	7.5
Water	22,058	1076	118	**187**	**9.2**

GCMF - Edge-intersection test

- The test has two phases:
  1) Test for MBR (built from edges) overlap.
  2) If edge pair passes the first test, we test for actual intersection test.
- Each pair is processed in a GPU-block.
  - Block-threads are synced.
  - using shared memory.
  - Load balanced within a GPU-block.



GCMF - Performance

- Up to 89 to 140 -fold speedup versus GEOS and PostGIS

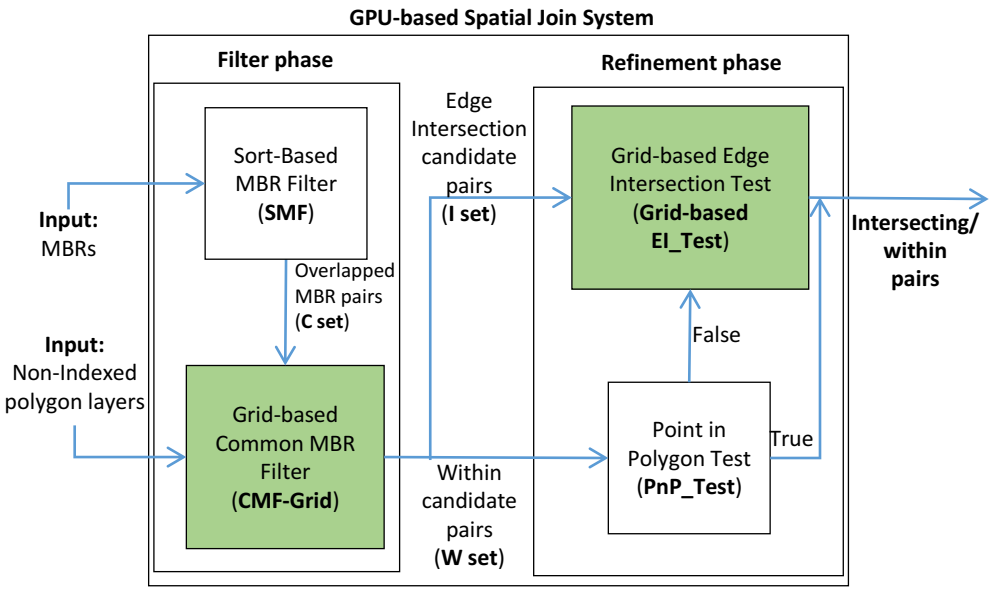
Dataset	Running Time (ms)			# of outputs
	PostGIS	GEOS	GCMF	
Urban	3,120	5,770	**52**	23,634
Telecom	17,8			581,351
Water	2			539,974

- ✓ Refinement portion reduced from more than 80% [Wang2014Haggis] to about 50%.
- ✓ Still too large!

			PnP	EI
Input	219,831 * 463,591	1,020,458	8	198,142
Reduction	> 99%	65%	46%	57%
Time Fraction	8.6%	21.27%	9.23%	**56.02%**

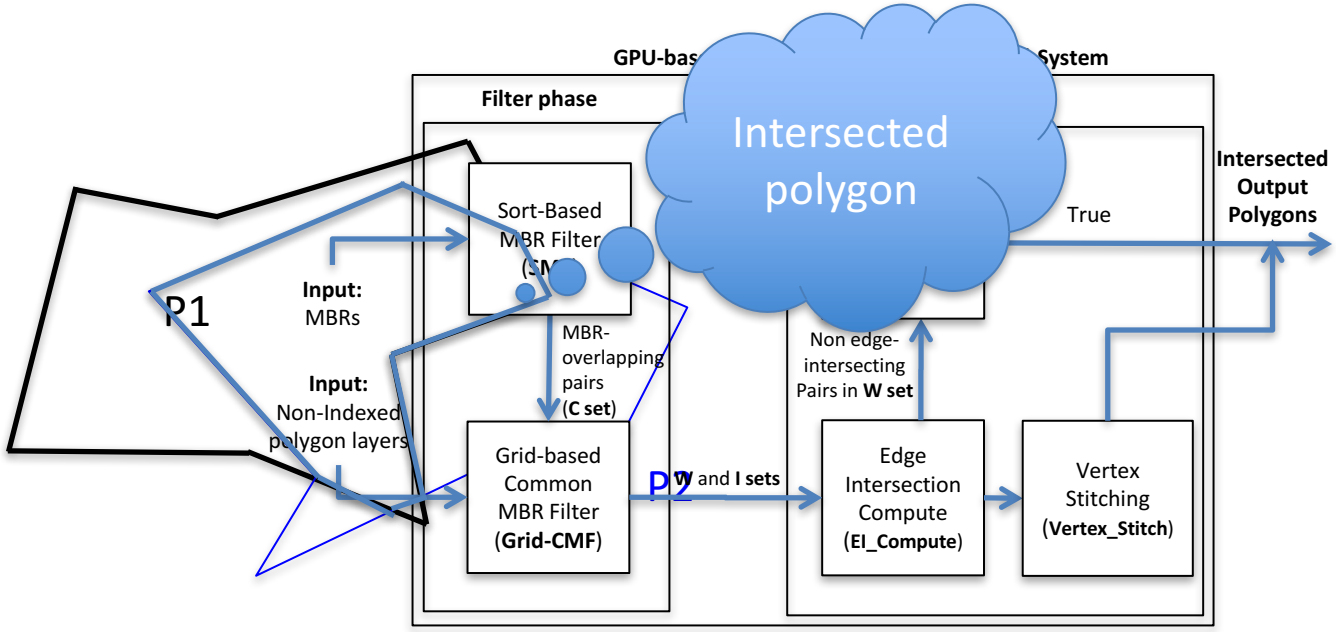
Proposed system 2: GCMF+

- ST_intersect predicate
- CMF is replaced by grid-CMF
- Grid-based edge-intersection test



Proposed system 3: Polygon Overlay

- Input: Two layers of polygons
- Output: Polygons generated from AND operation



Proposed system 4: MPI-Cuda-GIS

- **MPI-GIS**
  - Distributed homogeneous system based on MPI
  - Just CPU computing
  - No big data yet
- **MPI-Cuda-GIS**
  - Distributed Heterogeneous System
  - GPU-CPU computing
  - Handling big spatial data (Terabyte size)

Proposed system 4: MPI-Cuda-GIS

- Challenges:
  - IO:
    - Parallel IO libraries such as MPI-IO
    - Distributed database solutions: such kassandra
  - Scalability:
    - Keeping the same running time for larger datasets and more computing nodes
    - weak scalability condition
  - Load balancing/task partitioning

Conclusion

- Algorithms an data structures
  - Novel MBR overlap test algorithm
  - Novel linear time filters based on CMBR
- Spatial join processing systems
  - Single node, single GPU
  - Single node, multi GPU
  - Distributed heterogeneous nodes

Thanks!

?