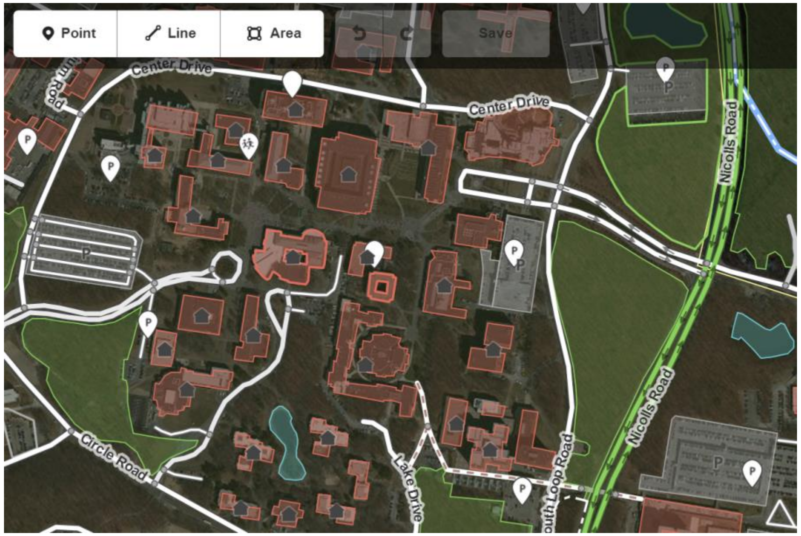
High Performance Spatial Queries and Analytics for Spatial Big Data

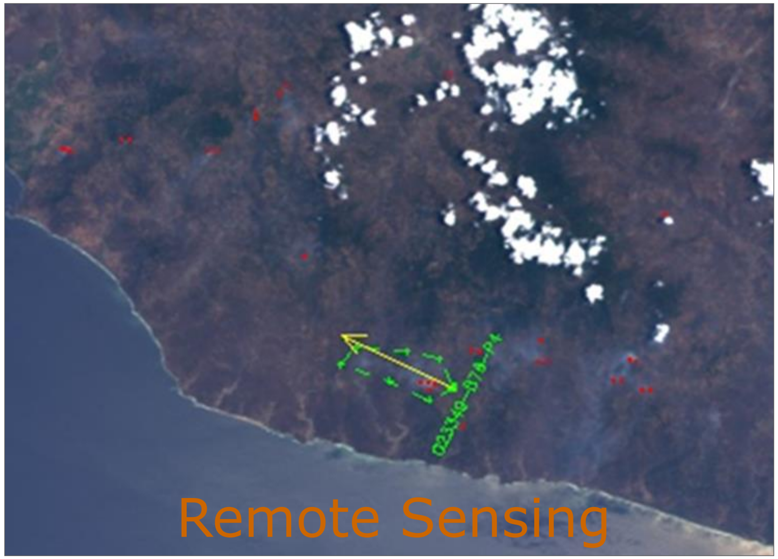
Fusheng Wang

Department of Biomedical Informatics
Emory University

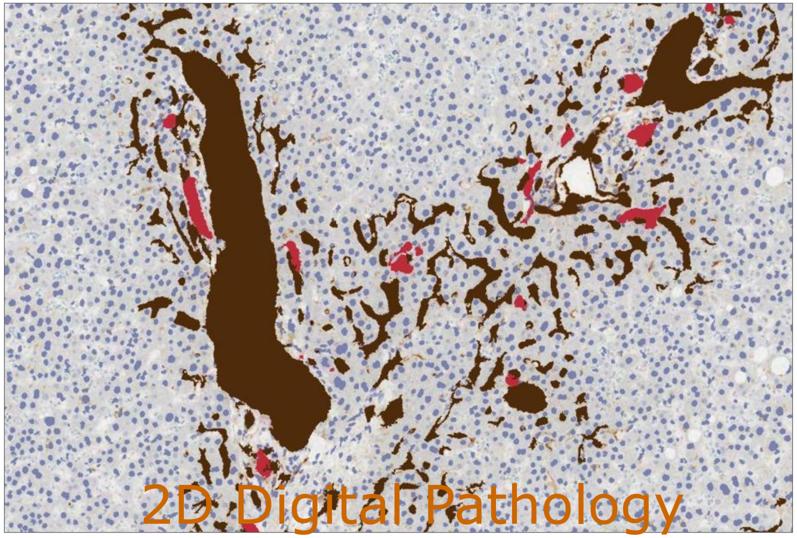
Spatial “Big Data”



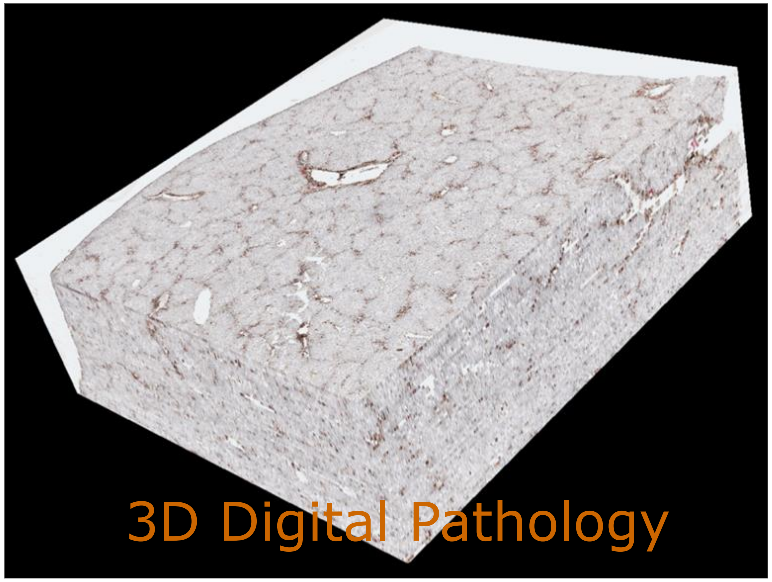
Geo-crowdsourcing:OpenStreetMap



Remote Sensing

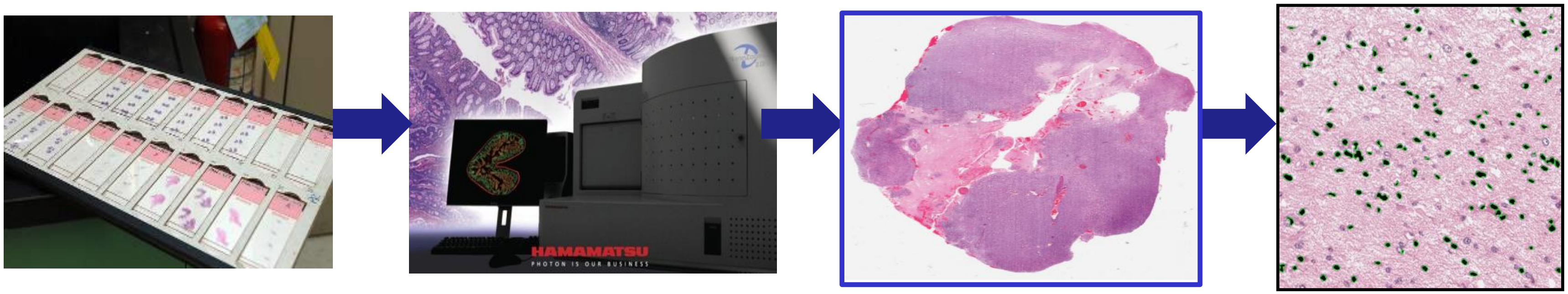


2D Digital Pathology



3D Digital Pathology

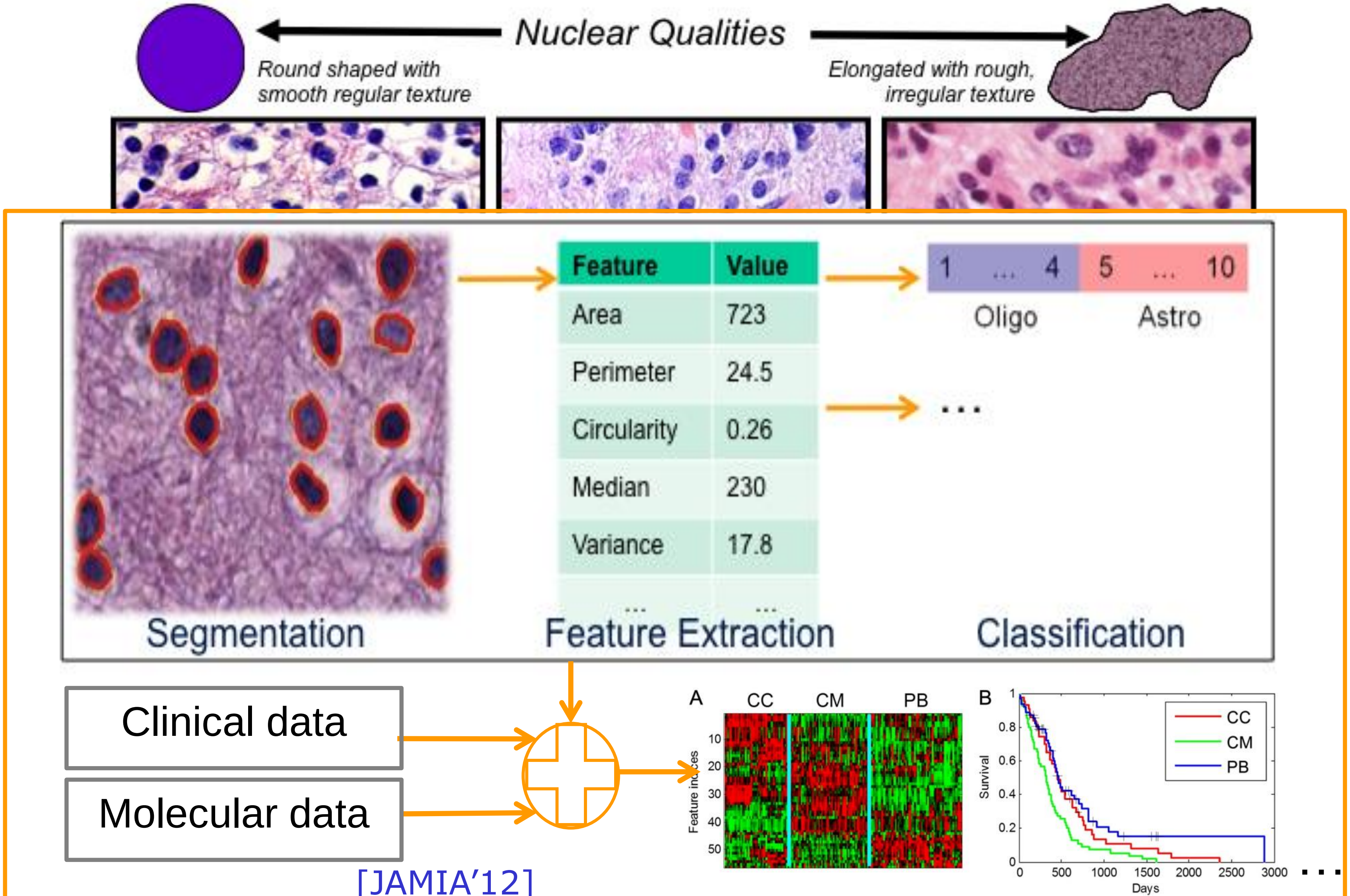
Pathology Analytical Imaging



Glass Slides | Scanning | Whole Slide Images | Image Analysis

- Provide rich information about morphological and functional characteristics of biological systems, have tremendous potential for understanding diseases and supporting diagnosis
  - e.g.: http://www.openpais.org/portal

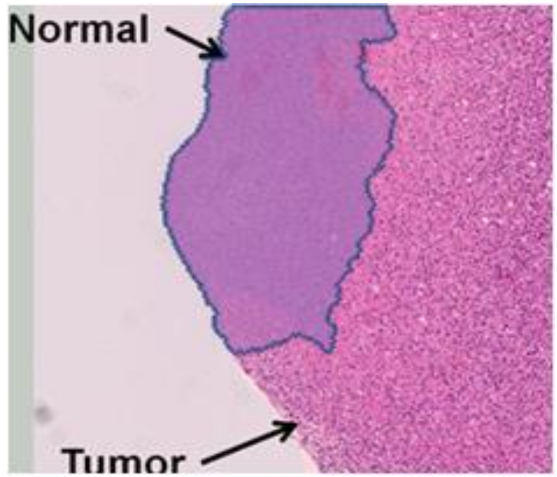
Example: Distinguishing Characteristics in Glioblastoma



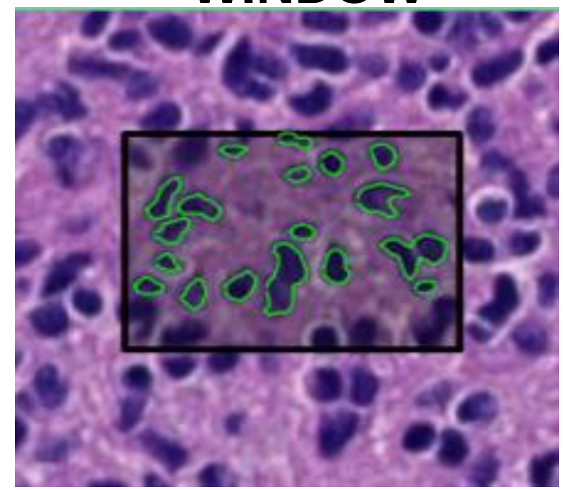
"GIS" Centric Queries

POINT

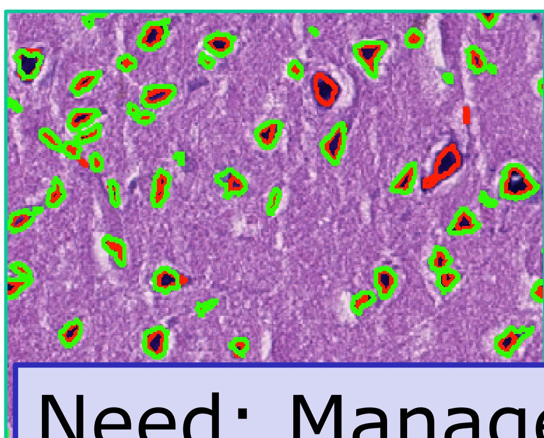
CONTAINMENT



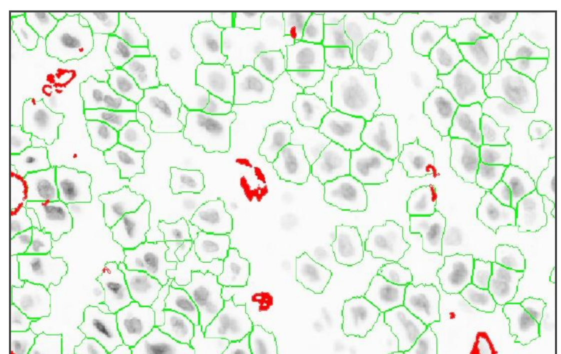
WINDOW



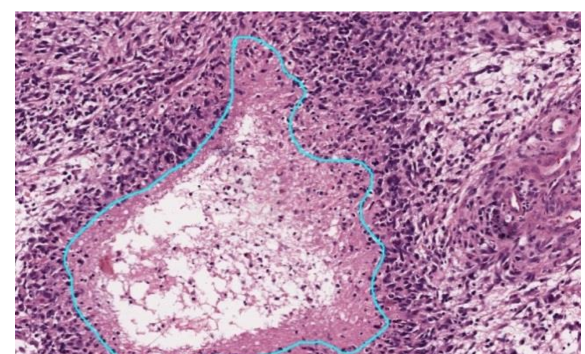
SPATIAL JOIN



NEAREST NEIGHBOR



DENSITY



Need: Manage, Query and Analyze Spatial Big Data

Spatial Queries and Analytics

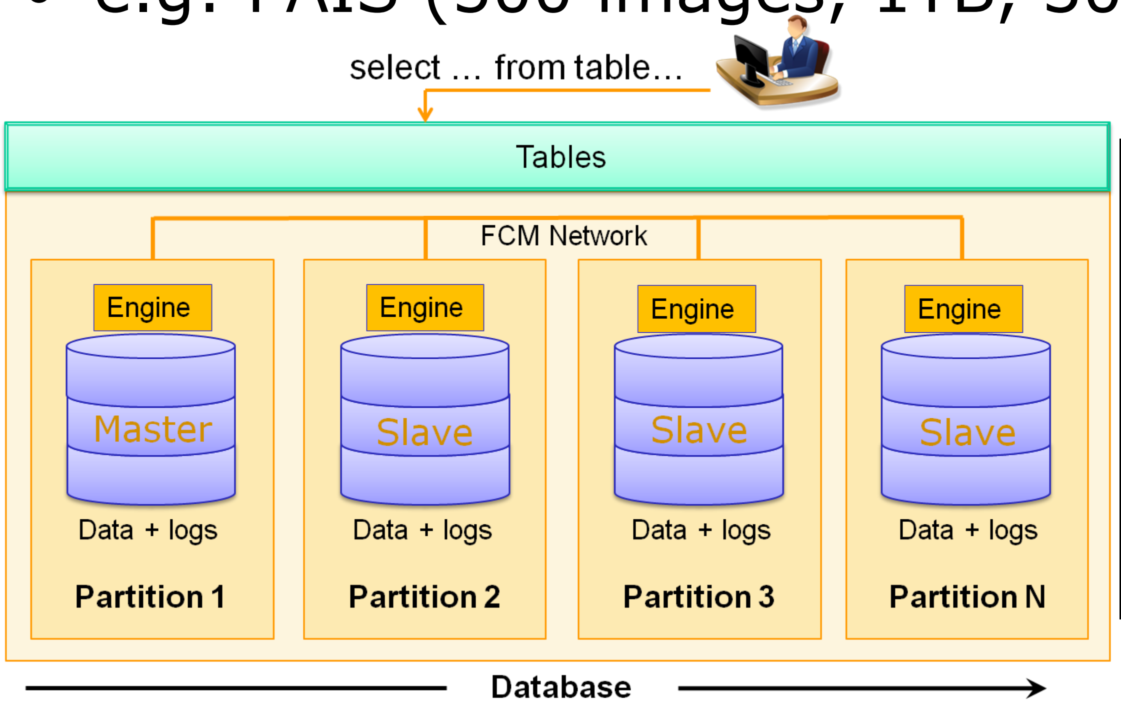
- Feature based descriptive queries
  - Feature based filtering or feature aggregation
- Spatial relationship based queries
  - Spatial join (two- or multi-), window, point-in-polygon
  - Polygon overlay or spatial cross-matching
- Distance based queries
  - Nearest neighbors
- Spatial analytics
  - Find spatial clusters, hotspots, and anomalies
  - Spatial relationship modeling, e.g., geographically weighted regression model (GWR)

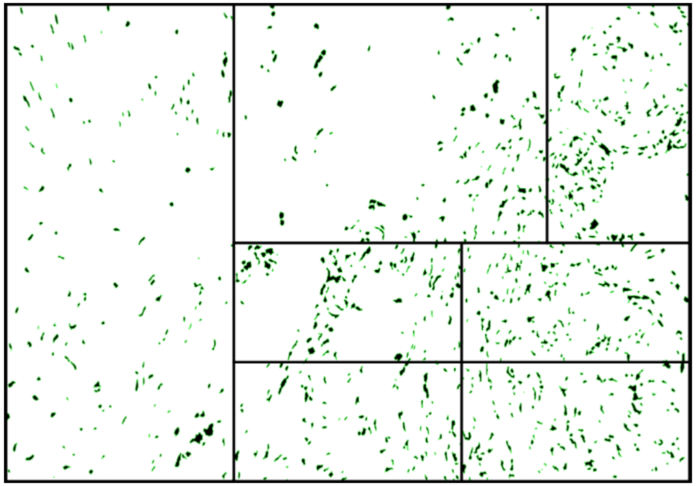
Requirements and Challenges

- Requirements: fast query response, and scalable and cost-effective architecture
- Explosion of derived data
  - 10^5x10^5 pixels per image
  - 1 million objects per image
  - Hundreds to thousands of images per study
  - OSM has two billion nodes, 600K contributors
- High computational complexity
  - Multi-dimensional
  - Spatial queries involve heavy duty geometric computations

Traditional Approach: Parallel SDBMS

- Shared nothing architecture through partitioning to increase I/O bandwidth via parallel data access
- Extended from ORDBMS with spatial data types and access methods
- Partitioning: even distribution of data and colocation
- e.g: PAIS (500 images, 1TB, 30 partitions)





[JPI'12, JPI'13]

EMORY UNIVERSITY

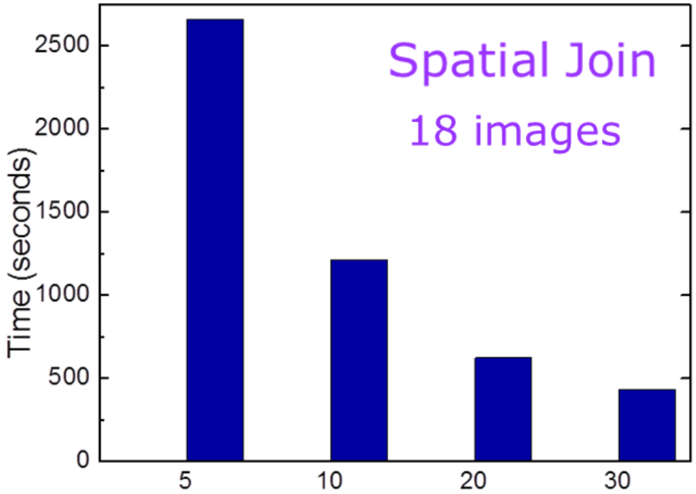
Advantages of Parallel SDBMS

- Comprehensive model and expressive query language for modeling and querying spatial data

Example spatial join query:

```
SELECT A.pais_uid, A.tilename, A.markup_id,
  CAST(db2gse.ST_Area(db2gse.ST_Intersection(a.polygon,b.polygon))/db2gse.ST_Area
  (db2gse.ST_Union( a.polygon, b.polygon)) AS DECIMAL(4,2)) AS area_ratio,
  CAST( db2gse.ST_Distance(db2gse.ST_Centroid(b.polygon),db2gse.ST_Centroid(a.polygon))
  AS DECIMAL(5,2) ) AS centroid_distance
FROM   pais.markup_polygon A, pais.markup_polygon B
WHERE  A.pais_uid ='oligoIII.2_20x_20x_NS-MORPH_1' AND
       A.tilename='oligoIII.2.ndpi-0000090112-0000024576' AND
       B.pais_uid ='oligoIII.2_20x_20x_NS-MORPH_2' AND
       B.tilename ='oligoIII.2.ndpi-0000090112-0000024576' AND
       db2gse.ST_Intersects(A.polygon, B.polygon) = 1;
```

- Scale out is possible



EMORY UNIVERSITY

Limitations of Parallel SDBMS

- The Cancer Genomics Atlas (TCGA):
  - 14,000 whole slide images, 30TB of results
  - Two months to get data loaded
  - 4 days to do a spatial join with 30 partitions
- Partitioning based scale out is possible but is very difficult and expensive to scale to many nodes
- DBMS not optimized for computational intensive operations
- Data loading is a major bottleneck
- Lack load balancing

Can we provide a solution with the advantages of RDBMS but is much more scalable and cost effective?

Goal: High Performance Spatial Queries and Analytics

- MapReduce provides a highly scalable and cost effective framework for processing massive data
  - Map step divides input into small problems by keys
  - Reduce step collects answers and combine them
- HDFS for fault tolerance and efficiency
- Spatial queries and analytics are intrinsically complex and difficult to fit into the model
- Hybrid CPU/GPU systems commonly available, but the capacity is often underutilized

There is a major step required on providing new spatial querying and analytical methods to run on such architectures

Our Approach: Hadoop-GIS

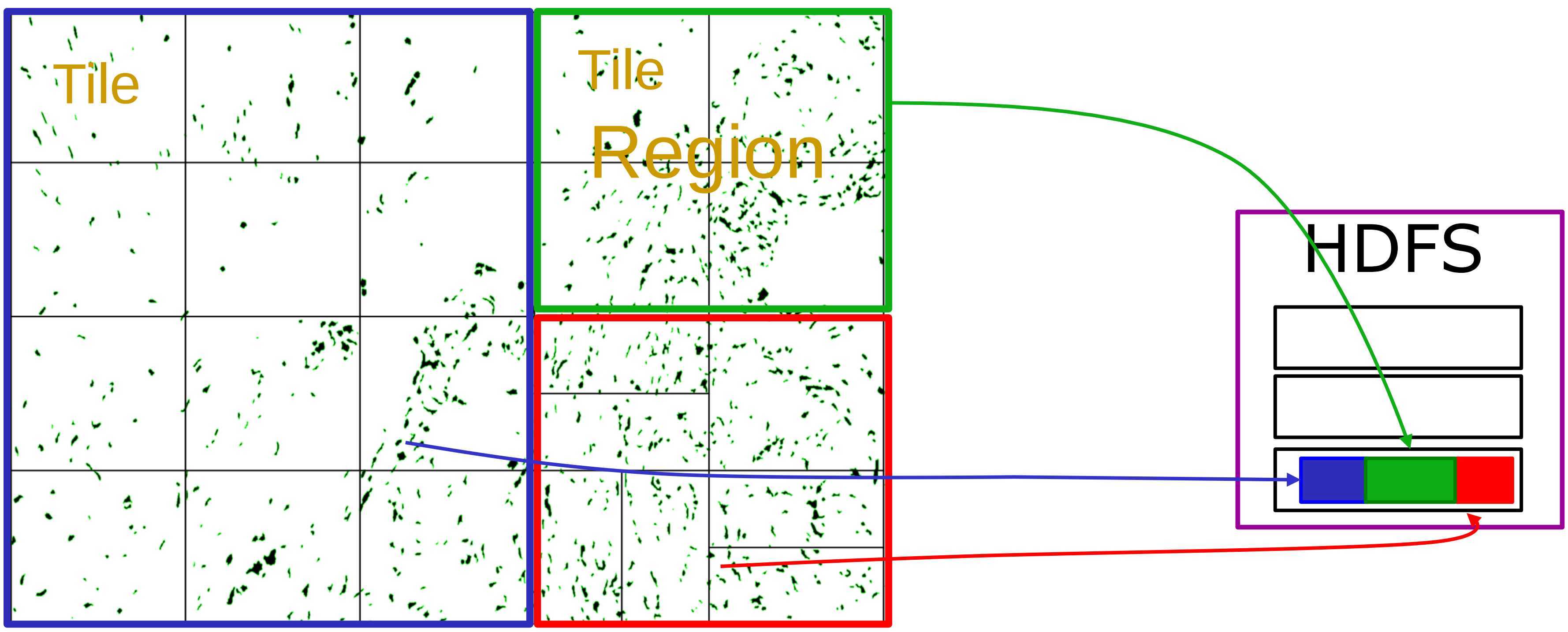
A general framework to support high performance spatial queries and analytics for spatial big data on MapReduce and CPU-GPU hybrid platforms

- Spatial data processing methods and pipelines with spatial partition level parallelism running on MapReduce
- Multi-level indexing methods to accelerate spatial data processing
- Query normalization methods for partitioning effect
- Declarative spatial queries and translation into MapReduce operations
- Utilize GPU to parallelize spatial operations and integrate them into MapReduce

[VLDB’12, GIS’12, GIS’13, VLDB’13]

Multi-level Spatial Indexing

- Tile indexing for managing tiles: filtering tasks in mapping stage
- Region based spatial indexing for grouping multiple neighboring tiles into a HDFS file: filtering data
- On demand indexing for intra-tile object queries



A General Framework of Spatial Data Processing in MapReduce

1. Spatial partitioning
2. Data staging and global indexing in HDFS

A. Block based filtering with region indexes

B. foreach *tile* in *input_region* **do** (in parallel)

   a. Tile based filtering based on tile indexes

   b. On-demand indexing for objects in the tile

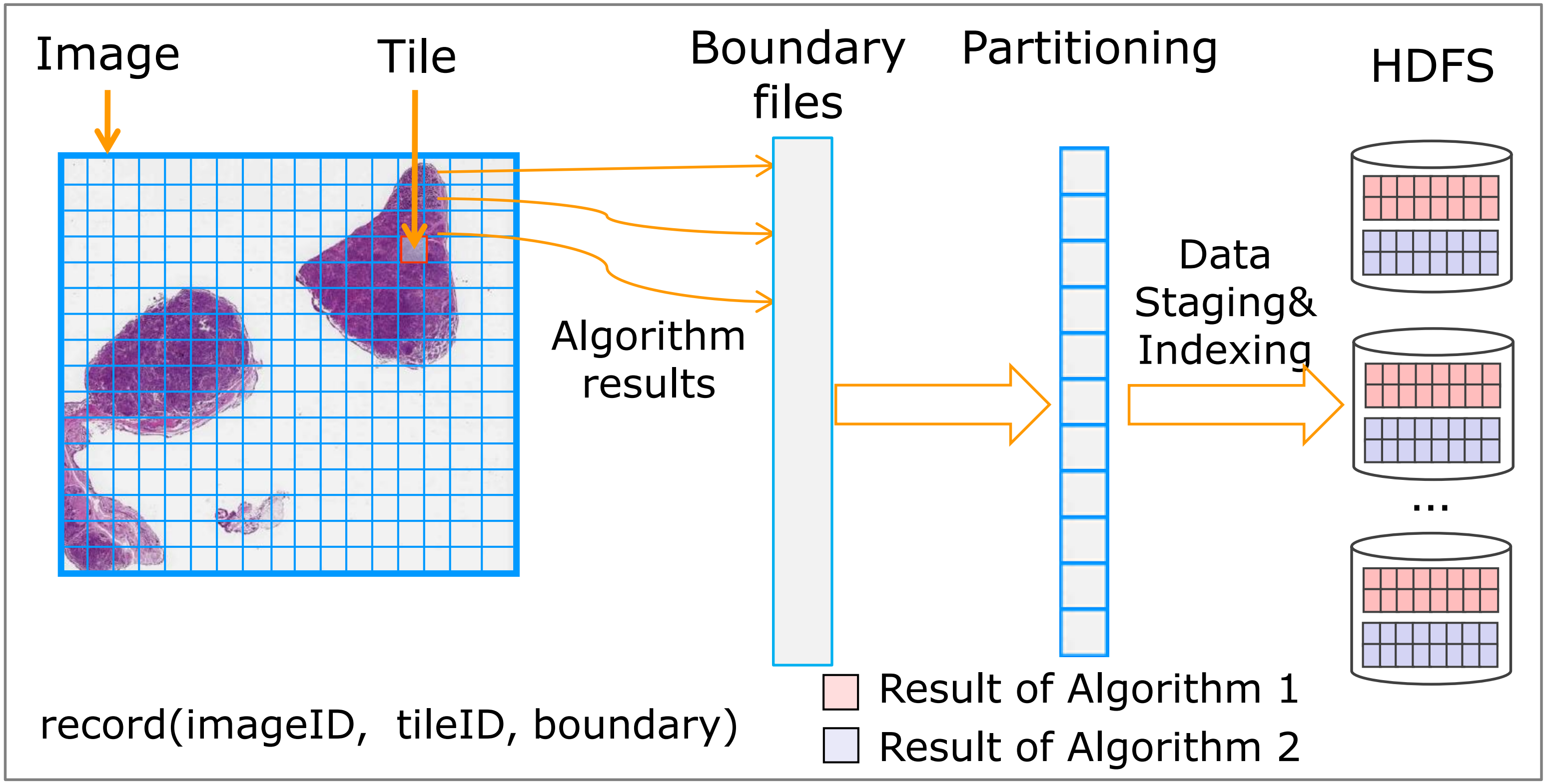
   c. Tile based spatial query processing

C. Boundary-crossing object handling

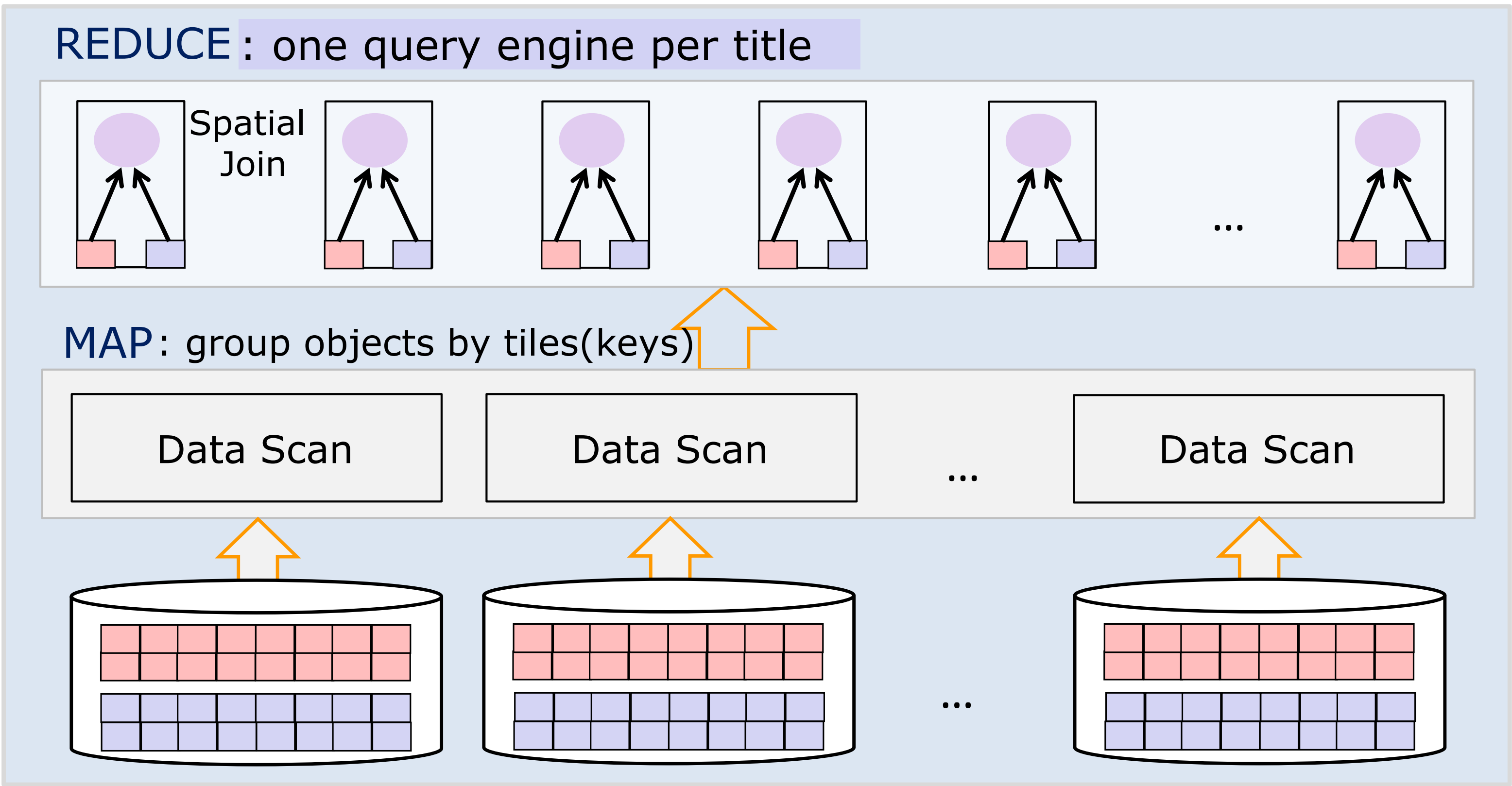
D. Post-query processing

E. Result storage on HDFS

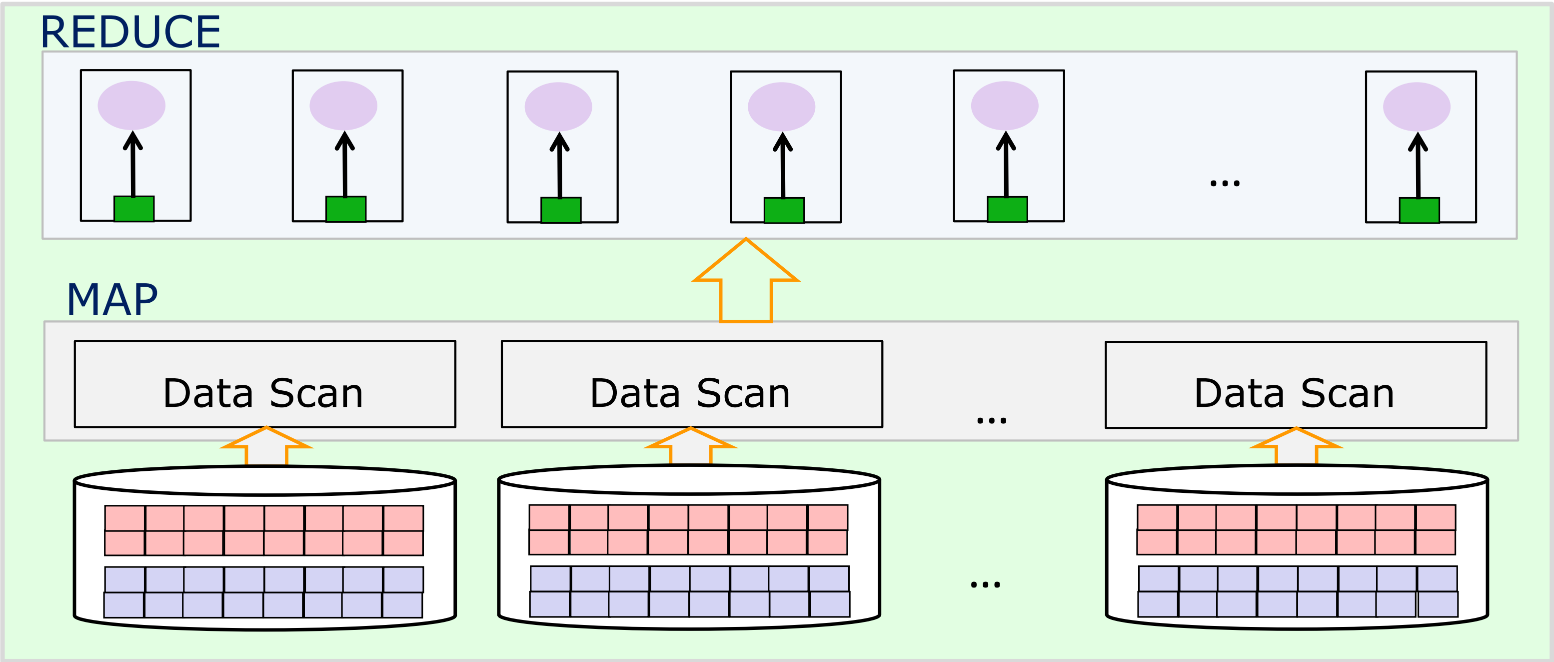
Example: Spatial Join in MapReduce: Data Staging and Indexing



Example: Spatial Join in MapReduce: Query Processing



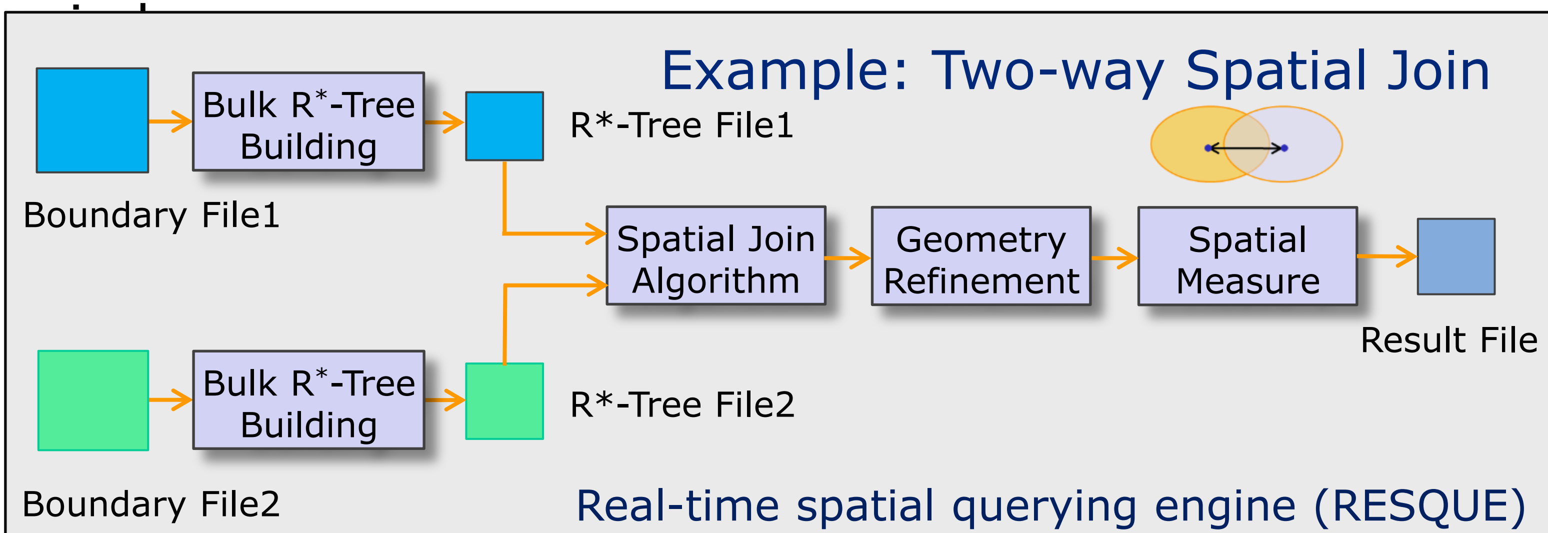
Example: Spatial Join in MapReduce: Result Normalization



EMORY UNIVERSITY

Real-Time Spatial Query Engine (RESQUE)

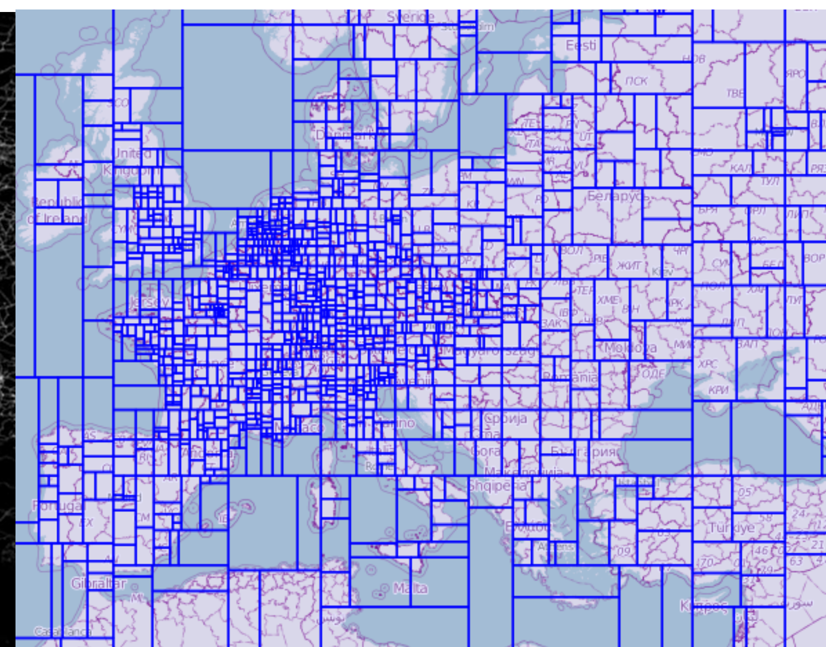
- Effective querying methods that can run in parallel in distributed computing environments
  - Spatial join, multi-way join, containment, nearest neighbor, and can be extended
  - Geometric computation library (GEOS)
- On-demand indexing based query processing
  - Mismatch between large block based storage and random



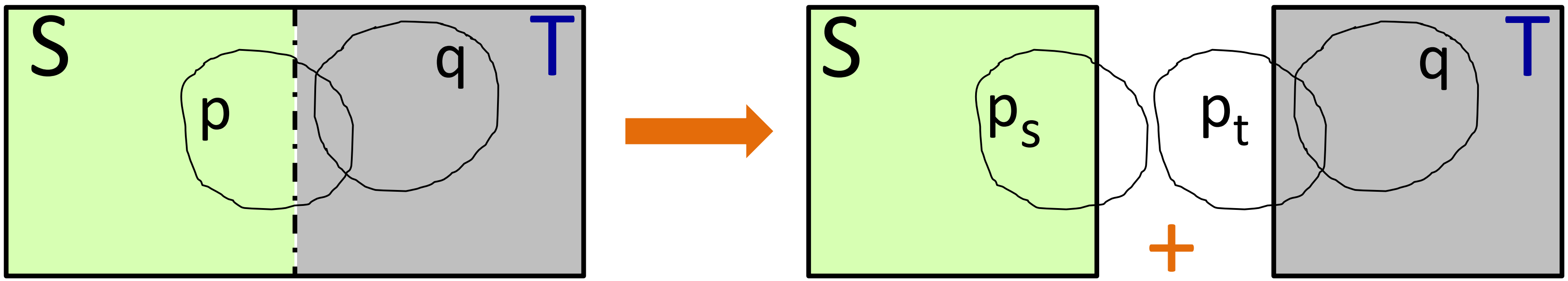
Real-time spatial querying engine (RESQUE)

Spatial Partitioning

- Effective partitioning is critical for task parallelization and load balancing: data skew
- Criteria: balanced distribution, granularity, overlapping
- Methods: top-down: recursive slicing; bottom-up: R-Tree packing
- Parallezation with MapReduce



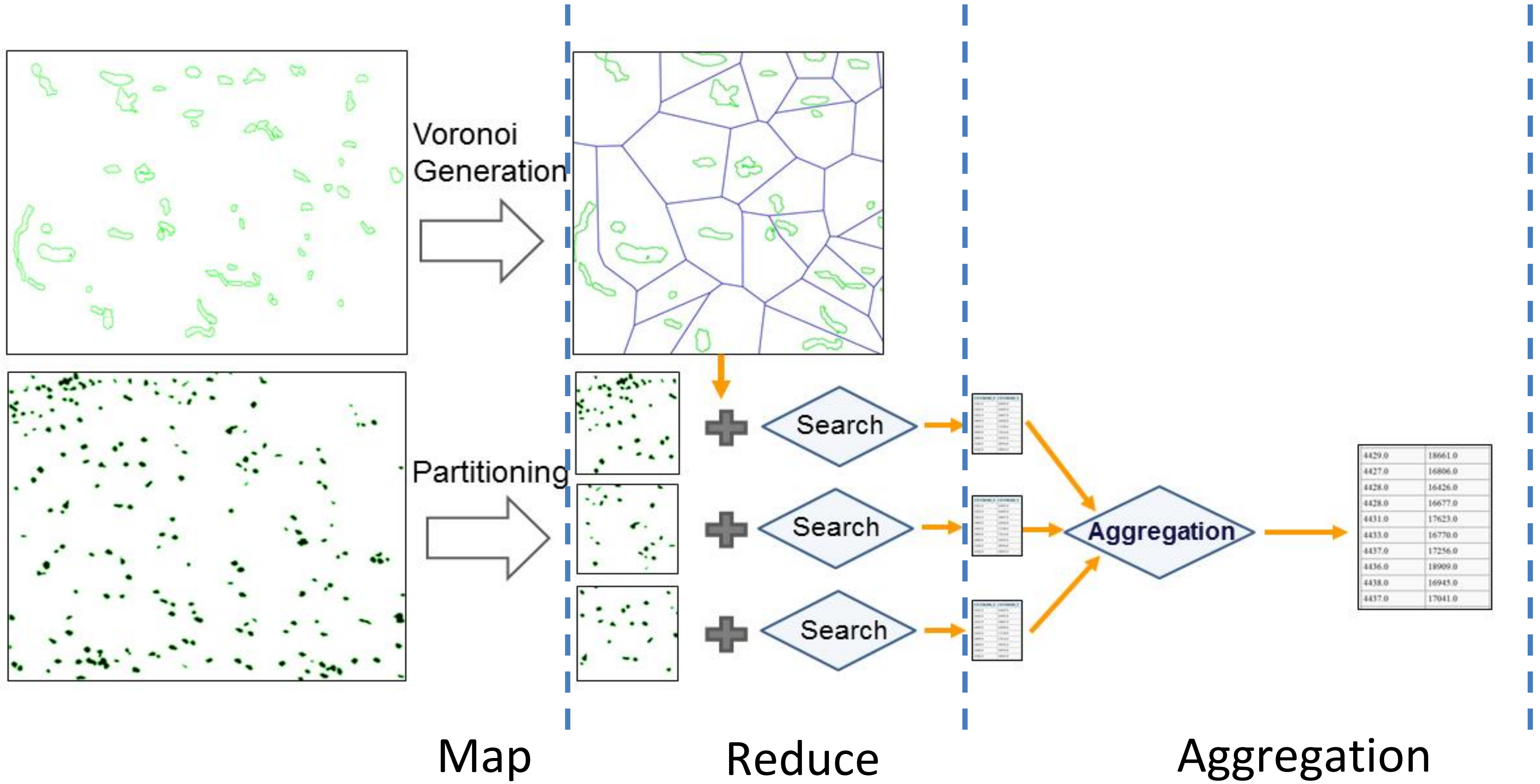
Boundary-Crossing Object Handling



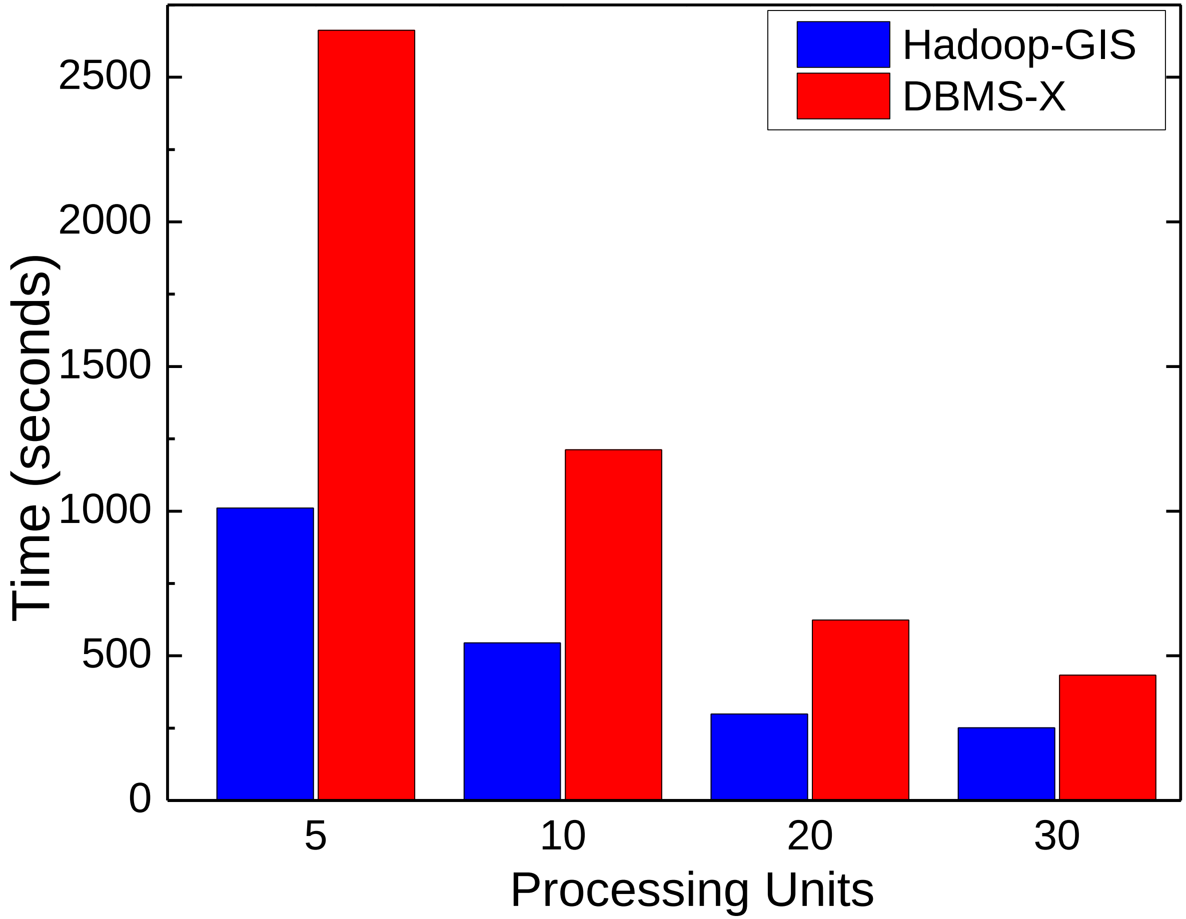
- “Multi-assignment, single-join”: replicate objects on boundaries to multiple tiles at partitioning
- Normalization methods are provided for each query type to correct answers
  - Spatial join: removal of duplicates
  - Voronoi diagram: reconstruction of diagrams on the boundaries

Nearest Neighbor Query Processing Workflow with MapReduce

- e.g.: for each cell find the closest blood vessel and return distance to that blood vessel
- Access methods can vary (R*-Tree, Voronoi, etc..)



System Performance: Hadoop-GIS vs Parallel Spatial SDBMS

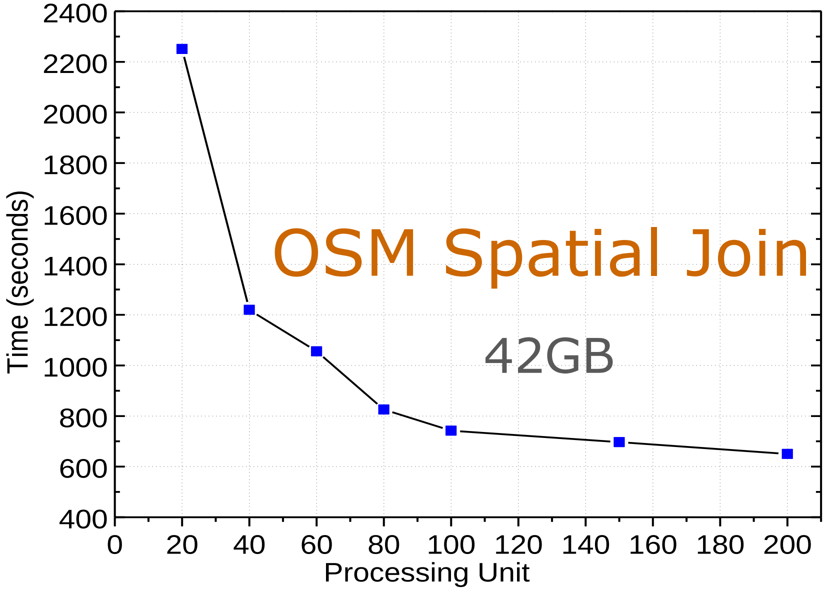


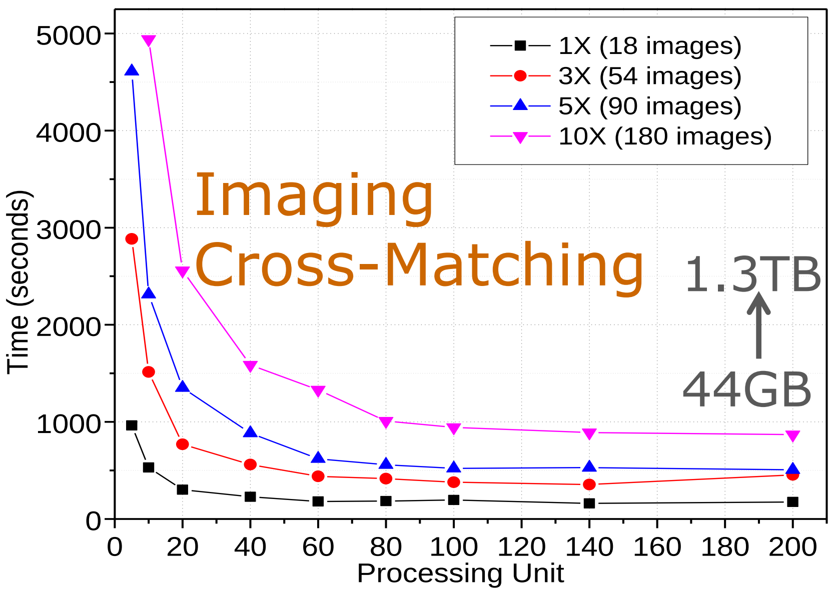
Spatial Join

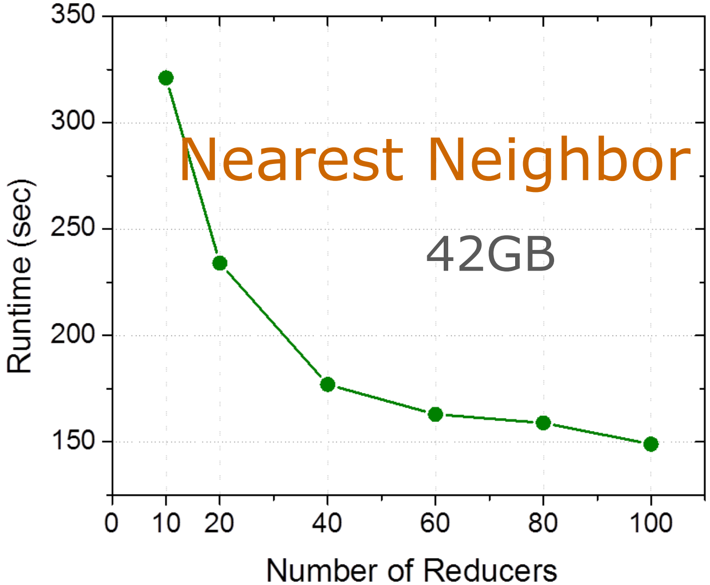
(boundary objects ignored)

EMORY UNIVERSITY

System Performance: Scalability







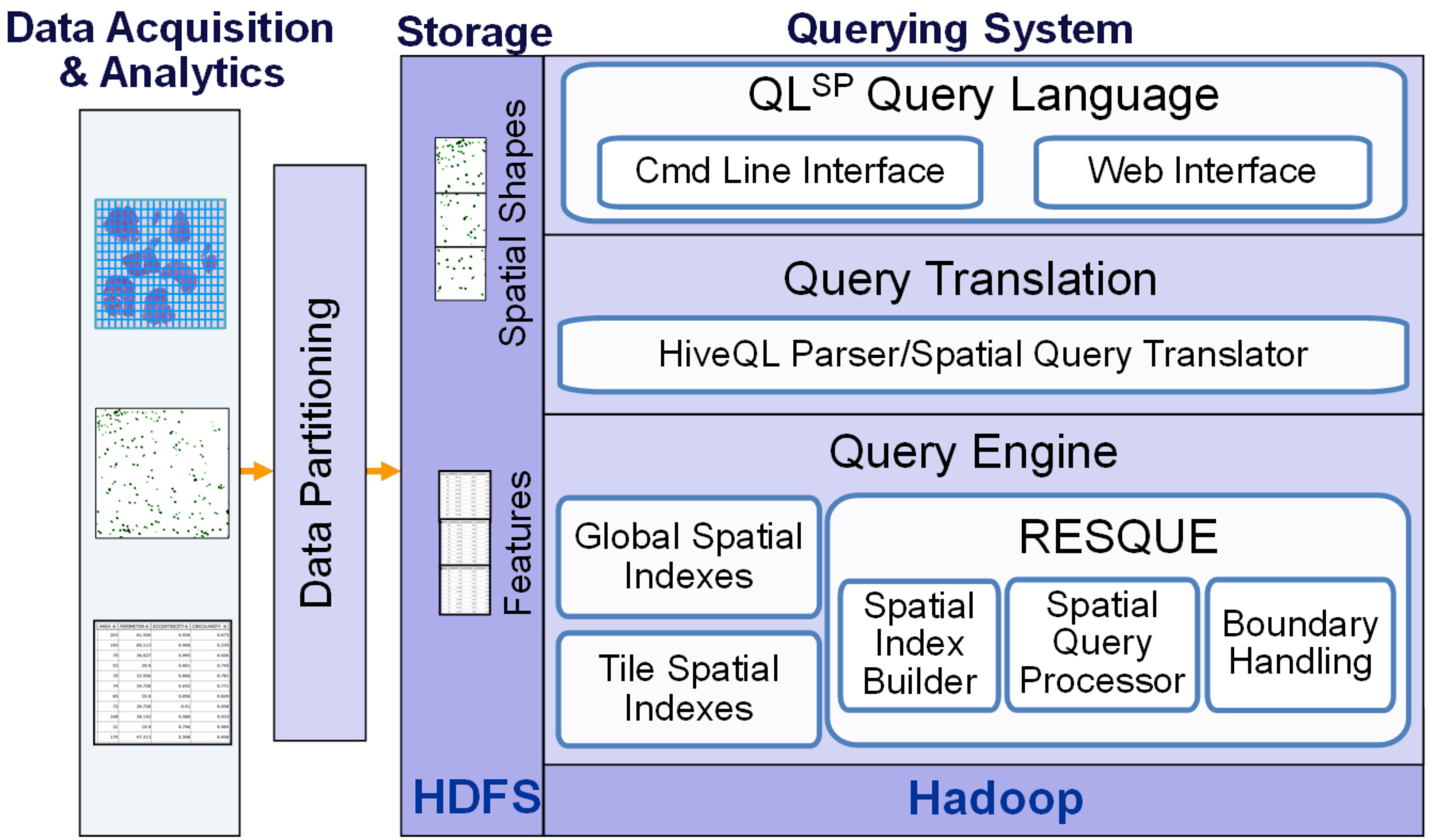
EMORY UNIVERSITY

Integration with Apache Hive

- Integrating declarative query languages with MapReduce is a major trend
  - Hive, Pig/Latin, Scope, Impala, Shark, Ysmart…
- Hive is a data warehouse infrastructure built on top of Hadoop, with a SQL like query language (QL)
- Hive provides major aggregation operations, and support user defined functions and data compression
- No spatial query support

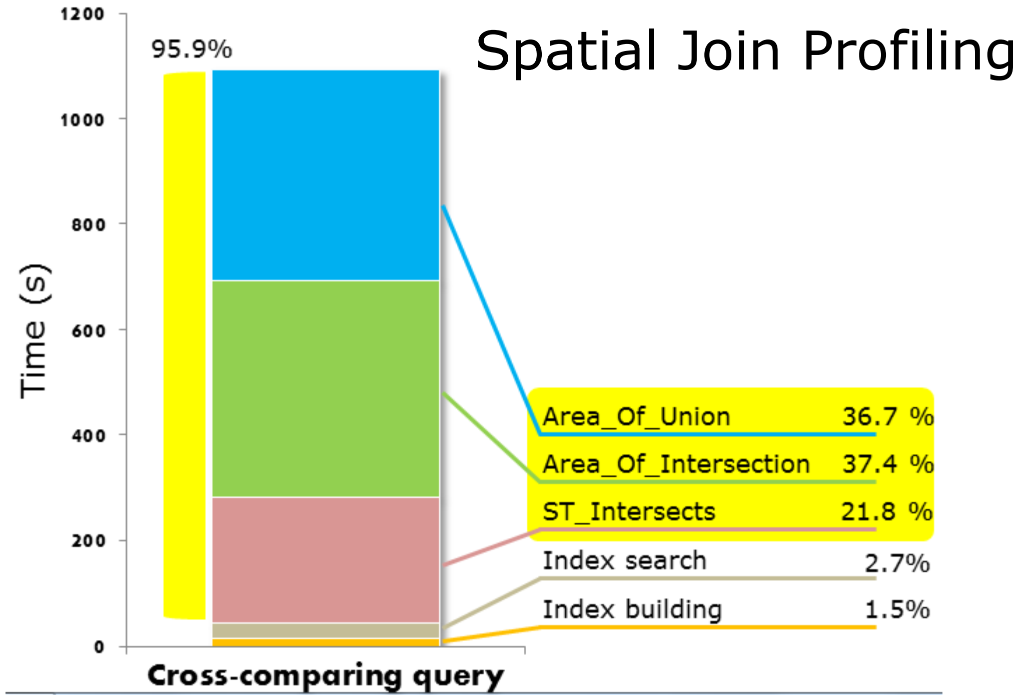
Goal: Integrate spatial data processing into Hive

Hadoop-GIS Architecture




```
SELECT
       ST_AREA(ST_INTERSECTION(ta.polygon,tb.polygon))/
       ST_AREA(ST_UNION(ta.polygon,tb.polygon))  AS ratio,
FROM   tcga_markups ta SPJOIN  tcga_markups ON
       (ST_INTERSECTS(ta.polygon, tb.polygon) = TRUE)
WHERE  ta.provenance='Algorithm1' AND
       tb.provenance='Algorithm2' ;
```

GPU Accelerated Spatial Operations



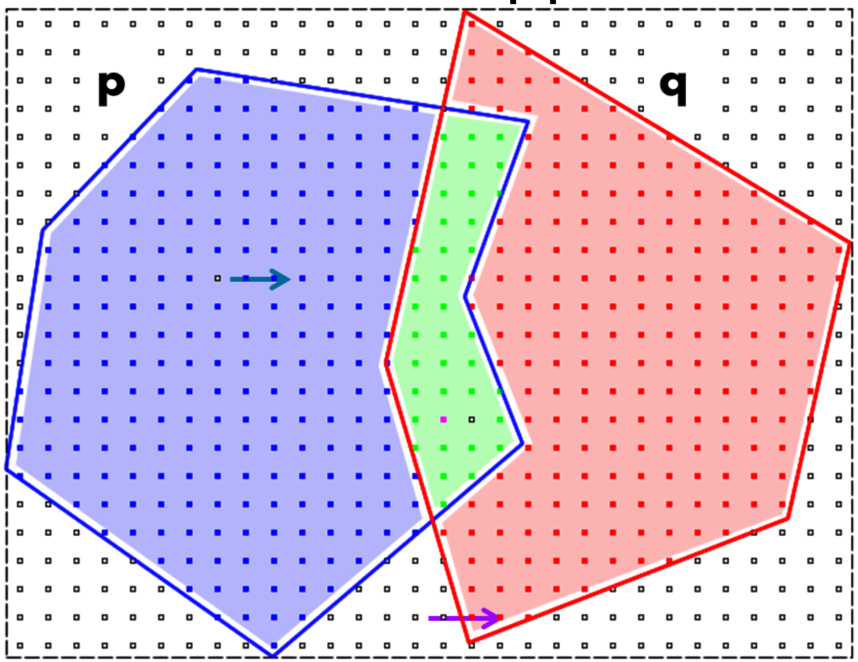
GPU Computing

- Massively parallel, hundreds to thousands of cores
- Cheap and highly available
- Programmable: CUDA, OpenCL

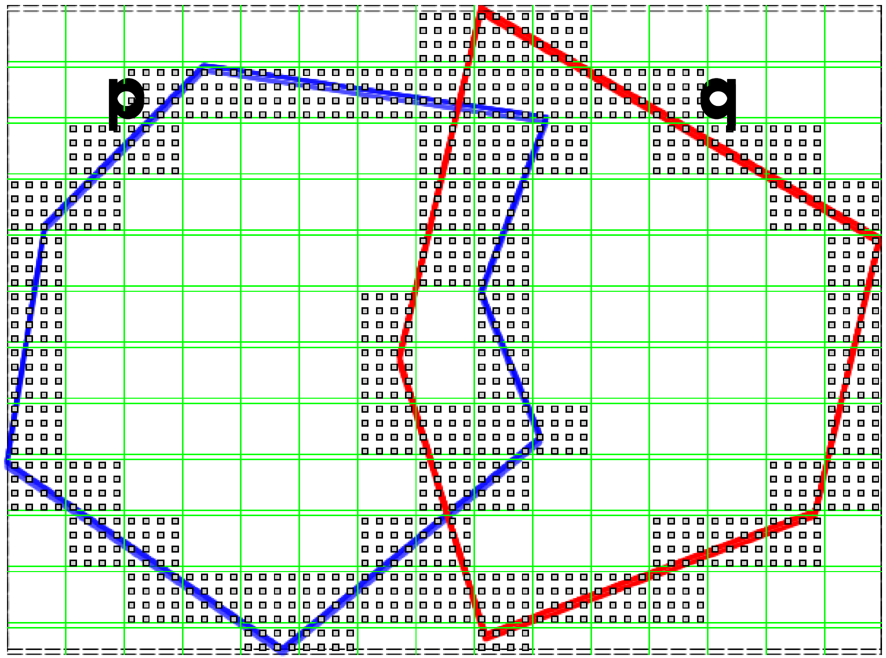
Goal: Exploit massive GPU parallelism and maximize data parallelism to accelerate spatial queries

PixelBox Algorithms for Spatial Cross-Matching

Monte-Carlo approach



PixelBox



- Speed up on cross-matching for one image on a single GPU (512 cores): 120X

[VLDB’12]

Ongoing Work (NSF CAREER)

- Create a high performance software system for spatial queries and analytics of spatial big data on MapReduce and CPU-GPU hybrid platforms
- Promote the use of the created open source software to support problem solving in multiple disciplines, and educate the next generation workforce in big data
  - Spatial and location based services (with Pitney Bowes)
  - 3D Imaging GIS (with Leeds, Emory)
  - Social media spatial analytics (twitter data)
  - Complex spatial analytics: spatial clustering, regression
  - Hybrid GPU-CPU processing

Thank you!

https://github.com/EmoryUniversity/libhadoopgis