Towards Building a High Performance Spatial Query System for Large-Scale Medical Imaging Data

Ablimit Aji, Fusheng Wang, Joel H. Saltz
Emory University

Presenter: Yandong Wang
“Big” Spatial Data

- Support of high performance queries on large volumes of scientific spatial data becomes increasingly important in scientific research.
- This growth is driven by not only geospatial problems but also emerging scientific applications that are increasingly data- and compute-intensive.
- For example, systematic analysis of large-scale pathology images generates tremendous amount of spatially derived quantifications of microanatomic objects.
Integrative Multi-Scale Biomedical Informatics

• Reproducible anatomic/functional characterization at gross level (Radiology) and fine level (Pathology)
• Integration of anatomic/functional characterization with multiple types of “omic”s information
• Create categories of jointly classified data to describe pathophysiology, predict prognosis and response to treatment

Emory In Silico Brain Tumor Research Center
Distinguishing Characteristics from Pathology Images

Nuclear Qualities
- Round shaped with smooth regular texture
- Elongated with rough, irregular texture

Oligodendrogloma → Nuclear Qualities → Astrocytoma

Clinical data + Molecular data

Segmentation
- Area: 723
- Perimeter: 24.5
- Circularity: 0.26
- Median: 230
- Variance: 17.8

Feature Extraction

Classification

[Graph showing progression over time]
Systematic Image Algorithm Evaluation

- High quality image analysis algorithms are essential to support biomedical research and diagnosis
  - Validate algorithms with human annotations
  - Compare and consolidate different algorithm results
  - Sensitivity study on algorithms’ parameters
- Example: What are the distances and overlap ratios between markup boundaries from two algorithms?

Cross match / join two spatial data sets

Need: Manage, Query and Compare Spatially Derived Information
Spatial Centric Queries

- **Point**
- **Containment**
- **Window**
- **Density**
- **SPATIAL JOIN**
- **Nearest Neighbor**
Parallel Spatial DBMS Approach

- Data loading takes very long time
- Limited scalability: possible but with high cost
- Expensive software/hardware license
- Limited query support
- Maintaining and tuning is complex

![Diagram of Parallel Spatial DBMS Approach]

Tables

- Engine
  - Master
    - Data + logs
  - Slave
    - Data + logs
  - Partition 1

- Engine
  - Slave
    - Data + logs
  - Partition 2

- Engine
  - Slave
    - Data + logs
  - Partition 3

- Engine
  - Slave
    - Data + logs
  - Partition N

Database

select ... from table...
Spatial Big Data Challenges

Both Data- and Compute-Intensive

• Explosion of derived data
  – $10^5 \times 10^5$ pixels per image
  – 1 million objects per image
  – Hundreds to thousands of images per study
  – **Big data demanding for high throughput**

• High computational complexity
  – Spatial queries include spatial refinement and spatial measurements, based on heavy duty geometric computation algorithms
  – **Demanding high performance**
High Performance Queries with MapReduce

• MapReduce is a parallel computing framework widely used for large-scale data analysis and queries
  – Widely used in major internet applications, open source version: Hadoop
  – Two simple UDFs for data processing: map() & reduce()
  – Very easy to develop scalable applications
  – Parallelization automatically managed by Hadoop

• Our approach:
  – Build efficient spatial query engine that can run easily deployed on clusters
  – Take advantage of MapReduce to run queries
High-Level Overview of Hadoop

- HDFS and the MapReduce Framework
  - Data processing with MapTasks and ReduceTasks
  - Three main steps of data movement.
- Data movement in the MapReduce framework is time-consuming
Data Movement in Hadoop MapReduce Framework

- Data processing pipeline consists of three phases: Map, Shuffle/Merge, and Reduce
- Lengthy shuffle/merge is caused by the AlltoAll global exchanging and merging of data segments
Spatial Join Processing with MapReduce

- **Staging:**
  - images are tiled into regular grids
  - tiles redistributed across HDFS as file blocks
  - small tiles are merged and metadata added into records

- **Map:**
  - identify records of same tiles to form tasks

- **Reduce:**
  - execute queries with real-time spatial query engine
  - aggregate query results
Real-Time Spatial Query Engine (RESQUE)

- Index building **on demand** (low overhead)
- Query pipelines to combine multiple steps of query processing
- Support of spatial join, multi-way spatial join, nearest neighbor, and highest density queries, and extensible for new ones
- Able to run in parallel with decoupled spatial processing in a distributed computing environment

**Example: Two-way Spatial Join**

- **Boundary File1** → **Bulk R*-Tree Building** → **R*-Tree File1** → **Spatial Join Algorithm** → **Geometry Refinement** → **Spatial Measure** → **Result File**

- **Boundary File2** → **Bulk R*-Tree Building** → **R*-Tree File2** → **Real-time spatial querying engine (RESQUE)**
Nearest Neighbor Query Processing Workflow

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

Map  Reduce  Reduce
System Architecture

- SQL query interface for ease of use
- Queries are translated into MapReduce with YSmart-Spatial
- Hadoop executes the translated MapReduce code
- Relies on the RESQUE for spatial query processing
Spatial Query Engine Performance

- R*-Tree building: ~16%, R*-Tree join: ~84%
- Scalability: multiple small spatial index has same performance as combined big index
- Storage: compression of boundaries in R*-Tree leave nodes as chain code: save 42% space
Scalability Experiments

• Experimental Setup
  – in-house cluster with 10 nodes (192 cores)
  – join query: a set of 18 images (0.5 M nuclei/image)
  – nearest neighbor query: 50 images from TCGA
  – data is stored and processed in vector format

• Join Query Types

  star join

  clique join
Star Join

- Choice of join algorithm
  - R*-Tree join
  - Partition based spatial merge join

![Graphs comparing R*-Tree join and PBSM join runtime](image-url)
Clique Join

R*-Tree join

PBSM join
Nearest Neighbor Query

R*-Tree

Voronoi Diagram

Runtime (sec)

Number of Reducers

Runtime (sec)

Number of Reducers

Image Partition

Tile Partition
Data Skew in Spatial Query Processing

- Skew in spatial data
- Hash partitioning is skew oblivious
- Query runtime is bounded by the longest running partition (straggler)
Query Optimization Methods

\[ Q = R \bigintersects S = \bigcup_{i=1}^{N} R_i \bigintersects S_i \]

\[ W_j = \sum_{i \in P_j} \text{Cost}(R_i \Join S_i) \]

\[ \text{Cost}(R_i \Join S_i) = \alpha |R_i| + \beta |S_i| + \gamma \]
Summary and Future Work

• Effective management of **big spatial data** is one of the pressing challenges for next generation integrative biomedical research.

• We propose and provide a high performance MapReduce based querying system for large scale spatial data.

• We empirically test the concept with analytical medical imaging as an example application.

• The system is efficient, cost effective, scalable and easy to use.

• We are planning to turbo-charge the system with GPU
Family of GIS (Updated)
Questions?

Hadoop-GIS:
https://web.cci.emory.edu/confluence/display/HadoopGIS