CG_Hadoop: computational geometry in MapReduce.
Ahmed Eldawy, Yuan Li, Mohamed F. Mokbel and Ravi Janardan
SIGSPATIAL’13.

Andres Calderon

October 18, 2016
Agenda

1. Background

2. Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3. Experiments

4. Conclusions
1 Background

2 Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3 Experiments

4 Conclusions
Hadoop is widely used on many applications...
  - Machine learning, Graph processing, Sorting...
But we are watching a big [spatial] data explosion...
  - satellites, smart phones, space telescopes, medical devices...
Computational geometry operations are quite costly.
- Convex hull - 4 billion points - up to three hours.
- Polygon union - 5 million polygons - around one hour.
SpatialHadoop

- Extend Hadoop for spatial analysis.
  - Efficient operation for spatial querying.
  - Support for spatial types (point, polygons, rectangle).
  - Spatial partitioning and indexing techniques (Grid, R-tree, R+-tree).
  - Open source (http://spatialhadoop.cs.umn.edu/).

(Eldawy and Mokbel, 2015)
Apply a divide and conquer approach.
Exploit spatial indexing and early pruning (SpatialHadoop).
Achieve up to 260x increase performance.
Part of the operation layer of SpatialHadoop.
Run under Hadoop, but not the same performance.
Main idea:

1. Partition the input.
2. Filter partitions that do not contribute to the answer when it is possible.
3. Apply the algorithm locally for each partition.
4. Merge the local answers to compute a global result.
Agenda

1. Background

2. Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3. Experiments

4. Conclusions
Agenda

1. Background

2. Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3. Experiments

4. Conclusions
The main idea of our Hadoop polygon union algorithm is to take advantage of a set of parallel machines, rather than performing all the work in a single machine, our proposed algorithm achieves orders of magnitude better performance than that of Hadoop. As each chunk is at most of size 64MB, the local union step mostly produces one output polygon, rather than a set of polygons as in Hadoop. Although the local and global union steps remain the same, they become much lighter. The local union step mostly produces one polygon over nodes. In particular, we use the R-tree indexing in the underlying index structure in SpatialHadoop to be able to distribute polygons over nodes. In particular, we use the R-tree indexing in SpatialHadoop, where the size of each R-tree node is 64MB, to dynamically assign each polygon to one node. The decision of which node belongs to which partition is completely taken by the Hadoop load file component, where it basically assigns polygons to nodes randomly. As a result, and as can be seen in the figure, some polygons randomly. As a result, and as can be seen in the figure, some polygons remain completely disjoint after computing the union. Then, all nodes send their output to a single machine which executes the final union function, which runs locally in each machine. After performing the local union, each machine ends up with a set of polygons that represent one list, and computes their union using the traditional in-memory algorithm. Each machine will end up with only few polygons, making it possible to do the union using the in-memory algorithm. As each chunk is at most of size 64MB, each machine computes the union of its own chunk using a traditional in-memory algorithm. Each machine then sends the union of all polygons assigned to this machine. The reduce function takes the output of all local unions, combines them into one on a slave node. In the final answer step, each machine computes the union of all polygons assigned to one node. This step is performed by the Hadoop reduce function, which runs on a single machine to compute the final answer as given in Figure 1(b). The partitioning step in SpatialHadoop is done in a spatially-aware manner, as given in Figure 3, where adjacent polygons are assigned to the same machine. The main reason here is that we utilize the R-tree indexing in SpatialHadoop making the whole algorithm significantly faster. The pseudocode for the polygon union algorithm in SpatialHadoop is exactly the same as that of Hadoop (Appendix A.1).
Traditional algorithm
Hadoop - Partition

Partition  Local  Global
Hadoop - Partition

Partition   Local   Global
Hadoop - Local union

Partition  Local  Global
Hadoop - Global union

Partition   Local   Global
Hadoop - Global union

Partition    Local    Global
SpatialHadoop - Partition

Partition  Local  Global
SpatialHadoop - Partition
SpatialHadoop - Local union

Partition  Local  Global
SpatialHadoop - Global union

Partition  Local  Global
SpatialHadoop - Global union

Partition  Local  Global
Agenda

1. Background

2. Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3. Experiments

4. Conclusions
Skyline
Divide and conquer approach
Hadoop - Partition

Partition  Local  Global
Partition  Local  Global
Hadoop - Local

Partition  Local  Global
Hadoop - Global

Partition    Local    Global
Hadoop - Global

Partition   Local   Global
Skyline in SpatialHadoop

Figure 4: Skyline in SpatialHadoop
SpatialHadoop - Partition

Partition  Pruning  Local  Global
SpatialHadoop - Partition

Partition Pruning Local Global
SpatialHadoop - Pruning

Partition  Pruning  Local  Global
SpatialHadoop - Pruning

Partition  Pruning  Local  Global
SpatialHadoop - Local

Partition  Pruning  Local  Global
SpatialHadoop - Local

Partition  Pruning  Local  Global
SpatialHadoop - Global

Partition  Pruning  Local  Global
SpatialHadoop - Global

Partition    Pruning    Local    Global
Agenda

1. Background

2. Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3. Experiments

4. Conclusions
Convex Hull

The Convex Hull is a geometric operation that finds the smallest convex polygon that contains a given set of points. In the left image, we have a set of random points. The right image shows the convex hull of these points, which is the boundary that encloses all the points with the minimum area.
Andrew’s Monotone Chain algorithm

https://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain
Hadoop - Partition

Partition  Local  Global
Hadoop - Partition

Partition  Local  Global
Hadoop - Partition

Partition  Local  Global
Hadoop - Local

Partition   Local   Global
Hadoop - Global

Partition  Local  Global
Given two pairs of cells $C_1 = h_c_1, c_2i$ and $C_2 = h_c_3, c_4i$, we apply the skyline algorithm four times to select the partitions.

Figure 5: Convex hull in SpatialHadoop

(a) Input
(b) Skyline max-max
(c) Skyline max-min
(d) Skyline min-min
(e) Skyline min-max
(f) Final answer
SpatialHadoop - Partition

Partition  Pruning  Local  Global
SpatialHadoop - Partition

Partition  Pruning  Local  Global
SpatialHadoop - Pruning

Partition  Pruning  Local  Global
SpatialHadoop - Pruning

Partition  Pruning  Local  Global
SpatialHadoop - Local

Partition  Pruning  Local  Global
SpatialHadoop - Global

Partition  Pruning  Local  Global
SpatialHadoop - Global

Partition  Pruning  Local  Global
Agenda

1. Background

2. Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3. Experiments

4. Conclusions
Farthest Pair
Rotating Calipers method

Hadoop - Partition

Partition  Local  Global
Hadoop - Local

Partition  Local  Global
Hadoop - Local

Partition  Local  Global
Hadoop - Local

Partition  Local  Global
Hadoop - Global
Hadoop - Global

Partition  Local  Global
Farthest pair in SpatialHadoop

Figure 6: Farthest pair algorithm in SpatialHadoop

(a) Min and Max distances
(b) Pruning rule
SpatialHadoop - Partition

Partition  Pruning  Local  Global
SpatialHadoop - Partition

Partition  Pruning  Local  Global
SpatialHadoop - Pruning

Partition  Pruning  Local  Global
SpatialHadoop - Pruning

Partition  Pruning  Local  Global
SpatialHadoop - Pruning

Partition  Pruning  Local  Global
SpatialHadoop - Local

Partition  Pruning  **Local**  Global
SpatialHadoop - Local

Partition  Pruning  Local  Global
SpatialHadoop - Global

Partition  Pruning  Local  Global
SpatialHadoop - Global

Partition  Pruning  Local  Global
Agenda

1. Background

2. Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3. Experiments

4. Conclusions
Closest Pair
Divide and conquer approach
Partition  Local  Global
Hadoop - Partition

Partition  Local  Global
Hadoop - Local

Partition   Local   Global
Hadoop - Global
Closest pair in SpatialHadoop

Figure 7: Closest Pair in SpatialHadoop
SpatialHadoop - Partition
SpatialHadoop - Partition

Partition  Local  Pruning  Global
SpatialHadoop - Local

Partition  Local  Pruning  Global
SpatialHadoop - Pruning

Partition  Local  **Pruning**  Global
SpatialHadoop - Global

Partition  Local  Pruning  Global
SpatialHadoop - Global

Partition  Local  Pruning  Global
Experiments

Agenda

1 Background

2 Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3 Experiments

4 Conclusions
Setup

- **Cluster of 25 nodes:**
  - HDD from 50GB to 200GB.
  - RAM from 2GB to 8GB.
  - Processors 2.2GHz to 3GHz

- **Single machine:**
  - HDD 2TB.
  - RAM 16GB.
  - Processor 3.4GHz.
Datasets

- Real datasets (from OpenStreetMap):
  - OSM1: 164M polygons, 80GB.
  - OSM2: 1.7B points, 52GB.

- Synthetic dataset:
  - SYNTH: 3.8B points, 128GB.
  - Five different distributions.

(a) Uniform  (b) Gaussian  (c) Correlated  (d) Anti-correlated  (e) Circular
Experiments

Union

![Graph showing the comparison between Single, Hadoop, and SHadoop in terms of time (min) vs. file size (OSM1)]

- **Single**
- **Hadoop**
- **SHadoop**

Time (min) increases with file size (250M, 1G, 4G, 10G) for all methods.
Skyline

Correlated

Anti-correlated
Convex hull

Experiments

Uniform

Gaussian
Closest/Farthest pair

**Farthest Pair**

- **Single**
- **SHadoop**

**Closest Pair**

- **Single**
- **SHadoop**

File size (GB)

Time (min)
Agenda

1. Background

2. Computational Geometry Operations
   - Union
   - Skyline
   - Convex Hull
   - Farthest Pair
   - Closest Pair

3. Experiments

4. Conclusions
This paper introduced CG_Hadoop as a scalable and efficient MapReduce library.

Focused on 5 fundamental computational geometry problems...
- Polygon union, Skyline, Convex hull, Farthest and Closest Pairs.

Provided versions for Apache Hadoop and SpatialHadoop systems.
Distributed approach speed up performance.
Spatial partitioning allows early pruning which make it even more efficient.
Achieve up to 29x and 260x better performance.
Future ideas

- Working on more complex operations, for example motion patterns.
- Explore ports to new distributed platforms such as Spark or Simba.
Thank you!!!

Do you have any question?