SPATIAL BIG DATA MANAGEMENT AND ANALYTICS ON CLOUD ENVIRONMENTS AND MODERN COMPUTING INFRASTRUCTURES

Apr. 26th, 2016
Hoang Vo
Agenda

1. Introduction to Big Data Analytics
2. Introduction to Spatial Data Analytics
3. Introduction to Cloud Computing and Modern Architecture
4. Data Skew & Spatial Partitioning
5. Spatial Partitioning Algorithms
6. Existing Systems
7. Example Use Cases
8. Conclusion and Future Work
9. Q & A
Big Data

Era of Big Data

Why Big Data?

• Data-Driven Businesses
  – Who uses it?
  – What can we do with it?

(SAS Analytics Institute)

*Organizations such as companies, governments* use data to:

– Understand customers’ needs
  • E.g. adjusting web sites, highlighted items, delivering services

– Improve operation efficiency
  • E.g. sharing data, managing inventory

– Make decisions
  • E.g. computing risks, determining profitability of investment

Big Data?

- We (people) also use data:
  - Understand the environment
    - E.g. weather forecasting,
  - Improve operation efficiency
    - E.g. path-finding to destination
  - Make decisions
    - E.g. planning vacations, retirement, savings and etc.

Internet of Things

http://www.3g.co.uk/PR/Feb2015/internet-of-things-everything-you-need-to-know.html
From Data to Big Data

Characteristics of Big Data:

3 V’s of Big Data
• Volume
• Velocity
• Variety

Additional V’s
• Volatility
• Veracity
• Validity

http://www.3g.co.uk/PR/Feb2015/internet-of-things-everything-you-need-to-know.html
http://www.commerce-experts.com/
From Data to Big Data

IBM Research Statistics:

**Volume**
- Scale of Data
- 6 billion people have cell phones
- 8.7 billion people have cell phones
- World population: 7 billion

**Velocity**
- Analysis of Streaming Data
- The New York Stock Exchange captures 1 TB of trade information during each trading session
- Modern cars have close to 100 sensors that monitor items such as fuel level and tire pressure

**Variety**
- Different Forms of Data
- 30 billion pieces of content are shared on Facebook every month
- 400 million tweets are sent per day by about 270 million monthly active users
- 420 million wearable, wireless health monitors
- 4 billion+ hours of video are watched on YouTube each month

**Veracity**
- Uncertainty of Data
- In one survey, 27% of respondents were unsure of how much of their data was inaccurate

**The Four V’s of Big Data**
- Volume
- Velocity
- Variety
- Veracity

As of 2011, the global size of data in healthcare was estimated to be
- 150 exabytes (1.6 billion gigabytes)

By 2014, it’s anticipated there will be
- 420 million wearable, wireless health monitors
- 4 billion+ hours of video are watched on YouTube each month
- 400 million tweets are sent per day by about 270 million monthly active users
- 420 million wearable, wireless health monitors
- 4 billion+ hours of video are watched on YouTube each month
- 400 million tweets are sent per day by about 270 million monthly active users

By 2015, 4.4 million IT jobs will be created globally to support big data, with 1.5 million in the United States

Sources: McKinsey Global Institute, Twitter, Census, Barometer, EMC, SAS, IBM, NIST, GAO

www.ibmbigdatahub.com/
Tackling Big Data

A generic approach?

*Divide and Conquer*

Purpose-driven classification – What do we want from Big Data?

www.esri.com
http://www.tagoras.com/learning20/
Spatial Big Data

"Easy to visualize"

Prevalent in many domains:

- Geographical Information Systems (GIS)
- Location Based Social Networks (LBSN)
- Neuroscience
- Medical imaging
- Astronomy
- Nuclear physics
- and etc.
Spatial (Big Data) Analytics

Spatial (Location) Correlation

Common analytical tasks:
• Data Warehousing
• Query Processing
• Machine Learning & Data Mining

E.g.:
• Pricing houses
• Analyzing social trends
Spatial (Big Data) Analytics

Challenges:
- Volume
- Velocity
- Variety

Opportunities:
- Improving storage
- Improving processing power

Cloud computing

https://geoross.wordpress.com/
http://oi-pro.com
What is Cloud Computing?

- Amazon AWS: "Cloud Computing", by definition, refers to the on-demand delivery of IT resources and applications via the Internet with pay-as-you-go pricing.

- Benefits of Cloud Computing
  - Service Availability
  - Data Availability
  - Massive economies of scale
  - Increased speed
  - And etc.

- Major providers
  - Amazon, Google, Microsoft

https://aws.amazon.com
Cloud Computing

Remember the Big Data characteristics

- Volume
- Velocity
- Variety

Opportunities:

- Improving storage:
  - Shared & Virtual Memory
  - Physical disk
- Encoding & Compression
- **Processing power**
  - *Cloud computing*
  - Parallelization

Modern algorithms operating on the spatial attributes:
Shared & Virtual Memory
Encoding & Compression
**Parallelization:** CPU & GPU

*Algorithms* are difficult to modify
Cloud Computing

Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...

SaaS
CRM, Email, virtual desktop, communication, games, ...

PaaS
Execution runtime, database, web server, development tools, ...

IaaS
Virtual machines, servers, storage, load balancers, network, ...

AWS

Google Cloud Platform

Google

Compute
Compute Engine

Storage
Cloud Data Storage
Cloud SQL

Networking
Load Balancing
Interconnect

Big Data
Cloud SQL
Data Flow

Services
Cloud DNS
Cloud Endpoints
Cloud Storage
Prediction
Cloud Computing and Modern Architecture

Another example of architecture

Hierarchy Abstraction

Apache Big Data Stack

https://www.researchgate.net
http://infomall.org
Cloud Computing and Modern Architecture

Horizontal Scalability/Parallelism
Spatial Analytics

Data Management and **Query Processing**

- Concept of Spatial Objects

- Example of a complex spatial query

  Find all houses available for sales that are larger than 1500 sq ft and are within 20 miles of a park or a river

**Divide and Conquer**

- Spatial Query Categories
  - Thematic
  - Geometric
  - Topological

https://msdn.microsoft.com
Cloud Computing and Modern Architecture

**Thematic** and **Geometric** queries can be handled by:

*Spatial operation libraries*
- CGAL, GEOS, OpenCV, PySAL, and etc.

**Architectures supporting record-level parallelism**
- MPI
- OpenMP
- MapReduce
- Spark
- Storm
- and etc.

- Other consideration:
  - GPU-based

[https://www.researchgate.net](https://www.researchgate.net)
Architectures / Platforms

• MPI
  - Distributed Computing Framework
  - Load Balancing and Granularity Issues

  ![MPI Diagram]

  - Sequential program on each core
  - Local data in each process
  - Explicit **Message Passing** by calling `MPI_Send` & `MPI_Recv`

• OpenMP:
  - Parallel Computing Framework
  - Explicit Parallelism

  ![OpenMP Diagram]

  - (shared data)
  - `some_serial_code`
  - `#pragma omp parallel for` for `j = ...; j++`
  - `block_to_be_parallelized`
  - `again_some_serial_code`
  - Master thread, other threads
  - ... sleeping ...
MapReduce

MapReduce parallel computing framework that is scalable and widely used for large-scale data analysis and queries

- Invented by Google
- Open source version: Hadoop
- Commercial version: Hortonworks, Cloudera

- Easy to develop scalable applications
  - Widely adopted by cloud environments and commodity clusters
- Simple programming model with two major steps:
  - *map* to partition problem into sub-problems
  - *reduce* to combine results
High Performance Computing with MapReduce

Key idea: Input (key, value) & Output (key, value)

Examples of many advantages:
- Resource and Task Scheduler
  - Load-balancing
- Easy-to-write
  - Users focus only on 2 functions
- Scalability
- Fault-tolerance
- Associated file system: HDFS

Disadvantages:
- I/O -- disk-based operations
Other Computing Architectures

**Spark**
- Generalizes MapReduce’s operations
- Uses Resilient Distribute Datasets (RDDs) and Data Frames
- RPCs for task dispatching and scheduling.
- Threadpool for scheduling instead of JVM container pool

**Storm**
- Real time streaming framework
- Topology: a directed acyclic graph
  - Spouts and Bolts

**Hybrid CPU/GPU based systems**
- System CUDA, OpenACC, and etc.
Common themes of parallel and distributed computing frameworks

How to select the right tool:

• Resource allocation: Scheduling & Load balancing
• Inter-process & Intra-process communication
• Resource utilization: CPUs & GPUs
• Batch-processing vs. Real-time processing
• Latency

Data Parallelism

• Job-level
• Task-level/process-level/Record-level
• Thread-level
Spatial Query Processing

**Thematic and Geometric Queries**
- Almost trivial integration into modern architectures

**Topological queries**

*Simplest solution:*
Divide the space into a grid of smaller subspaces

Topological query types
- Range (window)
- Spatial Join
- K-Nearest Neighbor

---

MapReduce Spatial Query Processing

A. Data/space partitioning;
B. Staging of partitioned data to HDFS;
C. Pre-query processing (optional);
D. for tile in input do
   Index building for objects in the tile;
   Tile based spatial querying processing;
E. Boundary object handling;
F. Post-query processing (optional);
Challenges in Spatial Partitioning

Simplest implementation of partitioning in MapReduce

- Parameters
- **Data Skew**
  - Size Distribution
  - Location Distribution
- Boundary Objects

(a) Fixed grid partition

OpenStreetMap

- Bin Width: 500
- Tile size: 1000x1000
- Tile object count avg: 993; stddev: 7,640
- Largest count of objects in a tile: 794,429
Solutions

Spatial Partitioning affects Spatial Query Performance

• Smarter Spatial Partitioning

• MapReduce and other platforms to perform partitioning

• Boundary Handling and Aggregation
What is Spatial Partitioning

(Wikipedia definition): **Spatial** or **Space partitioning** is the process of dividing a space (usually a Euclidean space) into two or more disjoint subsets (see also partition of a set). In other words, space partitioning divides a space into non-overlapping regions. Any point in the space can then be identified to lie in exactly one of the regions.

Spatial Big Data Management and Analytics:

- We relax the “disjoint” criterion
- These subsets must satisfy certain constraints
  - Computational tractability

Intuitively:

- Objects that are spatially closed together should be grouped into same partitions
- Partitions workload are similar and should be assigned equally to processing tasks.
Spatial Partitioning

“Simpler” partitioning problem:
Determining the non-overlapping rectangular partitioning with near-uniform data distribution within buckets is **NP-hard** (Muthukrishnan et al, 1999)

<table>
<thead>
<tr>
<th>4136</th>
<th>9340</th>
<th>9696</th>
<th>9824</th>
<th>1674</th>
</tr>
</thead>
<tbody>
<tr>
<td>6046</td>
<td>7655</td>
<td>453</td>
<td>5202</td>
<td>2977</td>
</tr>
<tr>
<td>8637</td>
<td>320</td>
<td>7717</td>
<td>601</td>
<td>7243</td>
</tr>
<tr>
<td>351</td>
<td>2843</td>
<td>660</td>
<td>8647</td>
<td>2507</td>
</tr>
<tr>
<td>8363</td>
<td>7622</td>
<td>4896</td>
<td>5847</td>
<td>8230</td>
</tr>
<tr>
<td>2256</td>
<td>9230</td>
<td>482</td>
<td>7978</td>
<td>129</td>
</tr>
<tr>
<td>1267</td>
<td>3630</td>
<td>177</td>
<td>2328</td>
<td>3842</td>
</tr>
<tr>
<td>7803</td>
<td>6673</td>
<td>5981</td>
<td>6911</td>
<td>4566</td>
</tr>
</tbody>
</table>
Input & Output Spatial Partitioning

- **Input:** Minimum Bounding Rectangles – why not centroids?
  
  Remember MapReduce processing of Spatial Join

- **Output:** Partition Schema/Layout/Index

Q1: Why are we not using **Histograms**?

Q2: Pre-indexed data set? Co-partitioning?
### Representative Partitioning Algorithms

Search for the **best** algorithm? One partitioning index or algorithms cannot fit all.

**Scalable, Efficient, and Generic**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Grid Partitioning</td>
<td>FG</td>
</tr>
<tr>
<td>Quadtree Partitioning</td>
<td>QT</td>
</tr>
<tr>
<td>Binary Space Partitioning</td>
<td>BSP</td>
</tr>
<tr>
<td>Hilbert Curve Partitioning</td>
<td>HC</td>
</tr>
<tr>
<td>Strip-based Partitioning</td>
<td>SLC</td>
</tr>
<tr>
<td>Boundary Optimized Space Partitioning</td>
<td>BOS</td>
</tr>
<tr>
<td>Sort Tile Recursive Partitioning</td>
<td>STR</td>
</tr>
</tbody>
</table>

**Classifications**

- Space-oriented vs. Data-oriented
- Top-down vs Bottom-up
- Overlapping & Non-overlapping
- Parallelizable vs. Non-parallelizable
Fixed Grid Partitioning

Simplest Algorithm
Space-oriented
Non-overlapping partitions

Algorithm 1: FixedGrid partition

\begin{algorithm}
\textbf{Input:} a set of spatial objects $R$
\textbf{Input:} partition capacity $c$

1. $m = \lceil \sqrt{|R|/c} \rceil$; \\
2. $U = \text{spatialUniverse}(R)$; \\
3. $G = \text{split } U \text{ into } m \text{ by } m \text{ grid}$; \\
4. \textbf{for } $r_i$ \textbf{in } $R$ \textbf{ do} \\
5. \hspace{1em} $g = \text{grids intersects with } r_i$; \\
6. \hspace{1em} assign $r_i$ to each grid in $g$; \\
7. \textbf{end}
\end{algorithm}
QuadTree

Uses a tree data structure in which each internal node has exactly four children of the same size (Can be built incrementally)

- Non-overlapping
- Space-oriented
Binary Space Partitioning

Uses a tree data structure in which each internal node has exactly two children (often not of same size). (Can be built incrementally)

- Non-overlapping
- Data-oriented
Hilbert Curve Partitioning

Hilbert Curve – one of Space Filling Curves

• Overlapping
• Data-oriented

Pseudo-Algorithm:
• Using Hilbert curve map the centroid of the spatial objects to obtain an ordering value
• Sort the Hilbert values.
• Each consecutive b objects are grouped together, and their spatial extents are used to form a spatial partition.
Strip-based Partitioning (SLC)

Resembles slicing a rectangle cake. Suitable for spatial join on trajectories

- Non-overlapping

---

Algorithm 3: Strip partition

\begin{verbatim}
Input: a set of spatial objects \( R \)
Input: partition capacity \( c \)
Input: partition dimension \( d \)

/* sort objects by mbr center in dimension \( d \) */

1 sort \((R,d)\);
2 \( U = \text{spatialUniverse}(R) \);
3 while \( R \) is not empty do
4     \( s = \text{cutStrip}(U, R, c) \);
5     while \( r_i \) intersects with \( s \) do
6         assign \( r_i \) to partition \( s \) ;
7         if \( s \) contains \( r_i \) then
8             remove \( r_i \) from \( R \);
9         end
10    end
11 end
\end{verbatim}
Boundary Optimized Space Partitioning

BOS makes greedy choice when producing the partitions
Optimizing:
*Percentage of boundary objects*

- Non-overlapping
- Data-oriented

**Algorithm 4: Boundary optimized strip partition**

```plaintext
Input: a set of spatial objects \( R \)
Input: partition capacity \( c \)
1. \( U = \text{spatialUniverse}(R) \);
2. while \( R \) is not empty do
   // cost in dimension \( x \)
   3. \( cx = \text{getCost}(U, R, c, x) \);
   // cost in dimension \( y \)
   4. \( cy = \text{getCost}(U, R, c, y) \);
   if \( cx \leq cy \) then
      // strip partition in \( x \) dimension;
   end
   else
      // strip partition in \( y \) dimension;
   end
3. end
```
Sort Tile Recursive Partitioning

- Overlapping
- Data-oriented

**Algorithm 5: STR Partition**

**Input:** a set of spatial objects $R$

**Input:** partition capacity $c$

1. $m = \lceil \sqrt{|R|/c} \rceil$
   
   // $m$ strips in dimension $x$

2. $S = \text{stripPartition}(R, x)$;

3. **for** $i \leftarrow \text{in } m$ **do**
   
   // $m$ strips in dimension $y$

4. $t = \text{stripPartition}(S[i], y)$;

5. **end**
Parallelization of Partitioning

- Parallel building of global index
  - Difficult to modify several algorithms
  - Not scalable, I/O and communication bottlenecks

- Multiple-step partitioning
  - Coarse-grained partitioning followed by a fine-grained partitioning.
    - Which methods are the best at each step?

- Sampling
  - What sampling rate is appropriate.
  - How to handle bad samples
Why so many tools?

Partitioning helps **Spatial Query Processing**

1) Which tool is best for which case?
2) How to set parameters to achieve best query performance?

**Query-driven Spatial Partitioning**

Optimization criteria:

- Minimize Query Runtime
- Maintain Tractability

Performance metrics for range query:

- *Query margin*

Performance for spatial join and k-NN:

- *Total partitioning and query runtime*
Spatial Query Processing Frameworks

- **RDBMS** – ArcGIS, Oracle, DB2, PostGIS
  - Limited partitioning capability
  - Limited flexibility
- **SpatialHadoop**
  - Non-dynamic partitioning
  - No boundary object handling

- **In-memory Join Systems**: TOUCH, SPINOJA, and etc.
  - Pre-indexed dataset
  - Lack of distributed support
Spatial Query Processing Frameworks

- RDBMS – ArcGIS, Oracle, DB2, PostGIS
  - Limited partitioning capability
  - Limited flexibility
  - UI/Visualization tool
- SpatialHadoop
  - Non-dynamic partitioning
  - No boundary object handling
- In-memory Join Systems: TOUCH, SPINOJA, and etc.
  - Pre-indexed dataset
  - Lack of distributed support
Spatial Query Processing Framework

- SCOPA and HadoopGIS
  - Scalable, efficient and generic framework
Other Use Cases

http://www.sciencealert.com/
http://earthobservatory.nasa.gov/Features/tracking/
http://www.stsconsultinggroup.com
Parameter Optimization

Simplified universal parameter for partitioning algorithms:

Number of partitions / Partition payload (number of objects per partitions)

\[
\alpha \quad \text{Partition determination cost per object} \\
\beta \quad \text{Boundary object factor} \\
\gamma \quad \text{Relative R-tree building cost per object pair} \\
\delta \quad \text{Total satisfied pair processing factor} \\
\omega \quad \text{MBB filtering factor} \\
\zeta \quad \text{Relative boundary handling cost factor}
\]

Assuming uniform distribution

\[
k_0 = \frac{2\gamma|R| + \omega(1 + \beta)|S| - \alpha(|R| + |S|)}{\delta \beta + \zeta}
\]

Heuristics:
Several sub-parameters are *invariant* when sampling data subsets.
Experiments & Results

AWS/ Amazon EC2

- 2 datasets: OpenStreetMap and Pathology Imaging (TCGA)
- 3 query types: containment, sp join and knn
Experiments & Results

- Partitioning Algorithm Runtime using a single-step partitioning:
Experiments & Results

• Spatial Join:

[Graphs showing runtime (min) vs. average partition payload for OSM and PI, with different algorithms represented by various symbols.]
Experiments & Results

• Nearest Neighbor

---

![Graph a) OSM](image1.png)

![Graph b) PI](image2.png)
Experiments & Results

- Simplified SCOPA’s Decision Table

- What is low, or medium, or high data skew?
  - Skew measures (not metric):
    - skewstdevratio
    - maxskewratio
Conclusion Future Work

- System Improvements:
  - Input and Application Integration
    - Not only text data is common
    - Non-text data sources:
      - Raster images
      - Whole-slide images (Pathology)
  - Example: pathology image processing

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>723</td>
</tr>
<tr>
<td>Perimeter</td>
<td>24.5</td>
</tr>
<tr>
<td>Circularity</td>
<td>0.26</td>
</tr>
<tr>
<td>Median</td>
<td>230</td>
</tr>
</tbody>
</table>

Step 1: Nuclei Segmentation

Step 2: Feature Extraction

Diagram: MapReduce Computing Framework Distributed File System

Whole Slide Images
  Tiling
  Tiled Images
  Compressed Images
  Segmentation
  Mask Images
  Boundary Vectorization
  Raw Polygons
  Boundary Normalization
  Normalized Polygons
  Isolated Buffer Polygon Removal
  Cleaned Polygons
  Non-boundary Polygons
  Boundary Polygons
  Whole Slide Polygon Aggregation
  Final Segmented Polygons
Conclusion Future Work

- **System Improvements:**
  - More dynamic and accurate query optimizer
    - Sensitivity analysis on system parameters
    - Maximize throughput while minimizing system resource utilization
  - MapReduce I/O improvement:
    - Combination of several steps
    - Spark
      - In-memory processing
      - In-memory partitioning
    - Storm
      - Streaming capabilities
  - Integration of GPUs
    - Hybrid scheduling and load balancing for GPUs and CPUs
Thank you!

Q & A