James Hughes

- Director of Open Source Programs at GA-CCRi
- GeoMesa core committer / product owner
- SFCurve project lead
- JTS committer
- Contributor to GeoTools and GeoServer

Emilio Lahr-Vivaz

- Technical Fellow at GA-CCRi
- GeoMesa core committer
Big Geospatial Data

- What type?
- What volume?
What Is Considered Big?

**Problem:** How do we define “big” geospatial data?
What Is Considered Big?

Problem: How do we define “big” geospatial data?

First refinement: What type of data do we are interested in?

Vector

Raster

Point Cloud
What Is Considered Big?

**Problem:** How do we define “big” geospatial data?

**First refinement:** What type of data do we are interested in?

- **Vector**
- **Raster**
- **Point Cloud**
What Is Considered Big?

Problem: How do we define “big” vector geospatial data?

Second refinement: How much data is “big”? What is an example?

GDELT: Global Database of Event, Language, and Tone

“The GDELT Event Database records over 300 categories of physical activities around the world, from riots and protests to peace appeals and diplomatic exchanges, georeferenced to the city or mountaintop, across the entire planet dating back to January 1, 1979 and updated every 15 minutes.”

~225-250 million records
What Is Considered Big?

**Problem:** How do we handle “big” vector geospatial data?

**Second refinement:** How much data is “big”? What is an example?

Open Street Map:

OpenStreetMap is a collaborative project to create a free editable map of the world. The geodata underlying the map is considered the primary output of the project.

Entire history can fit on a thumb drive.
Problem: How do we handle “big” vector geospatial data?

Second refinement: How much data is “big”? What is an example?

**What Is Considered Big?**

AIS / ADS-B / Mobility data

- **AIS** is a signal broadcast by maritime ships
- **ADS-B** is a signal broadcast by airplanes
- **Mobility data** is gathered by cell phone providers

IoT-based sources of spatio-temporal data can produce millions to billions of records per day!
What Is Considered Big?

**Problem:** How do we handle millions to billions of rows of vector data (typically points) arriving daily?
LocationTech
GeoMesa
Overview
What is GeoMesa?

A suite of tools for streaming, persisting, managing, and analyzing spatio-temporal data at scale.
What is GeoMesa?

A suite of tools for **streaming**, **persisting**, managing, and analyzing spatio-temporal data at scale
What is GeoMesa?

A suite of tools for streaming, **persisting**, managing, and analyzing spatio-temporal data at scale.
What is GeoMesa?

A suite of tools for streaming, persisting, managing, and analyzing spatio-temporal data at scale.
What is GeoMesa?

A suite of tools for streaming, **persisting**, managing, and **analyzing** spatio-temporal data at scale.
Database Vs Data Lake
What Databases Are We Discussing?

- In this context, we’re talking about distributed, NoSQL databases
  - Relational databases generally don’t scale well to the size needed
- Examples:
  - Apache HBase
  - Apache Accumulo
  - Apache Cassandra
  - Redis
  - Google Cloud Bigtable
What is a Data Lake?

- In this context, we’re talking about structured and semi-structured files hosted in cloud storage.
- Example of cloud storage:
  - AWS S3
  - Azure Blob storage
  - Google Cloud storage
  - HDFS
- Examples of file formats:
  - Apache Parquet
  - Apache Orc
  - JSON
  - CSV
Choosing the Database
Advantages of Using a Database

- **Speed**
  - Very fast to insert and retrieve data
  - Key-based design allows efficient updates

- **Data management**

- **Support for multiple query indices**
  - Accelerate typical query patterns

- **Powerful distributed processing**
  - HBase and Accumulo only
  - Heatmaps, summary statistics, fine-grained filtering, relational transforms
Detour - The Power of Space-Filling Curves

- NoSQL databases only support a single dimension, generally a sorted key-value index
- Spatial data has 2+ dimensions
- Enter GeoMesa and space-filling curves
The Power of Space Filling Curves (in one slide!)

- Goal: Index 2+ dimensional data
- Approach: Use Space Filling Curves
The Power of Space Filling Curves (in one slide!)

- Goal: Index 2+ dimensional data
- Approach: Use Space Filling Curves
- First, ‘grid’ the data space into bins
The Power of Space Filling Curves (in one slide!)

- **Goal:** Index 2+ dimensional data
- **Approach:** Use Space Filling Curves
- First, ‘grid’ the data space into bins
- Next, order the grid cells with a space filling curve
  - Label the grid cells by the order that the curve visits them
  - Associate the data in that grid cell with a byte representation of the label
The Power of Space Filling Curves (in one slide!)

- **Goal:** Index 2+ dimensional data
- **Approach:** Use Space Filling Curves
- **First,** ‘grid’ the data space into bins
- **Next,** order the grid cells with a space filling curve
  - Label the grid cells by the order that the curve visits them
  - Associate the data in that grid cell with a byte representation of the label
- **We prefer “good” space filling curves:**
  - Want recursive curves and locality
The Power of Space Filling Curves (in one slide!)

- Goal: Index 2+ dimensional data
- Approach: Use Space Filling Curves
- First, ‘grid’ the data space into bins
- Next, order the grid cells with a space filling curve
  - Label the grid cells by the order that the curve visits them
  - Associate the data in that grid cell with a byte representation of the label
- We prefer “good” space filling curves:
  - Want recursive curves and locality
- Space filling curves have higher dimensional analogs
Disadvantages of Using a Database

- Expensive to run
  - Compute is always on, even if not being used
- Complex to manage
- Storage can be expensive
  - Can be mitigated with HBase or Accumulo on S3 or Blob storage
    - But that adds to the complexity of managing the database
Choosing the Data Lake
Advantages of Using a Data Lake

- **Cost**
  - Cloud-native storage solutions are inexpensive
  - Data can be stored very compactly
  - Compute only needed on demand
- **Less complexity**
  - Simpler to manage
- **Throughput**
  - Very efficient to load massive datasets through an engine like Spark
- **Powerful distributed processing**
  - Can run massively parallel compute jobs
Detour - ETL vs ELT

- GeoMesa supports ETL of spatial data into Parquet or Orc files in cloud-native storage
  - Space-filling curves can be used to partition the data on disk
  - Columnar file formats can prune, transform and filter at query time
- GeoMesa also supports ELT of existing data
  - Not as performant but can operate on most common data formats
Spatial File Formats in GeoMesa

- No native spatial types
  - Geometries are built up with primitive columns
- Points
  - Stored as two columns of type Double, one for X and one for Y
  - Allows for push-down filtering against each dimension
- LineStrings, MultiPoints
  - Stored as two columns of type List[Double]
- MultiLineStrings, Polygons
  - Stored as two double precision List[List[Double]] columns
- MultiPolygons
  - Stored as two double precision List[List[List[Double]]] columns
Disadvantages of Using a Data Lake

- Data management
  - Can be hard to update records
  - Can be hard to organize your data
- Does not support different query patterns
- Is not very efficient with small targeted queries
Why
Not
Both?
Lambda Architecture

- Recent data can be stored in a database for performant querying
  - Fixed size keeps cost down
  - GeoMesa partitioned tables makes age-off trivial
- Historical data can be stored in a data lake for efficient analysis
Reference Architecture
Thanks!

- jhughes@ccri.com
- https://www.geomesa.org/
- https://gitter.im/locationtech/geomesa
- https://github.com/locationtech/geomesa
- Twitter @CCR_inc

CCRi is hiring!

https://www.ccri.com/careers/

- DevOps
- Software Engineers
- Data Scientists
Backup Slides
Visualization Example: Heatmaps

Without powerful visualization options, big data is big nonsense.

Consider this view of shipping in the Mediterranean sea.
Visualization Example: Heatmaps

Without powerful visualization options, big data is big nonsense.

Consider this view of shipping in the Mediterranean sea.
Generating Heatmaps

Heatmaps are implemented in DensityScan.

For the scan, we set up a 2D grid array representing the pixels to be displayed. On the region/tablet servers, each feature increments the count of any cells intersecting its geometry. The resulting grid is returned as a serialized array of 64-bit integers, minimizing the data transfer back to the client.

The client process merges the grids from each scan range, then normalizes the data to produce an image.

Since less data is transmitted, heatmaps are generally faster.
**Statistical Queries**

We support a flexible stats API that includes **counts**, **min/max** values, **enumerations**, **top-k** (StreamSummary), **frequency** (CountMinSketch), **histograms** and **descriptive statistics**. We use well-known streaming algorithms backed by data structures that can be serialized and merged together.

Statistical queries are implemented in [StatsScan](#).

On the region/tablet servers, we set up the data structure and then add each feature as we scan. The client receives the serialized stats, merges them together, and displays them as either JSON or a Stat instance that can be accessed programmatically.
Filtering and transforming records overview

Using Accumulo iterators and HBase filters, it is possible to *filter* and *map* over the key-values pairs scanned.

This will let us apply fine-grained spatial filtering, filter by secondary predicates, and implement projections.
Pushing down filters

Let’s consider a query for tankers which are inside a bounding box for a given time period.

GeoMesa’s Z3 index is designed to provide a set of key ranges to scan which will cover the spatio-temporal range.

Additional information such as the vessel type is part of the value.

Using server-side programming, we can teach Accumulo and HBase how to understand the records and filter out undesirable records.

This reduces network traffic and distributes the work.
Projection

To handle projections in a query, Accumulo Iterators and HBase Filters can change the returned key-value pairs.

Changing the key is a bad idea.

Changing the value allows for GeoMesa to return a subset of the columns that a user is requesting.
Row vs Columnar Layouts

- **Row layout**
  - All the data for a single *record* is contiguous
  - Easier to write and stream

- **Columnar layout**
  - All the data for a single *column* is contiguous
  - Can be compressed much more efficiently
  - Requires much less I/O for filtering and projections
### Row vs Columnar Layouts

<table>
<thead>
<tr>
<th>session_id</th>
<th>timestamp</th>
<th>source_ip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1331246351</td>
<td>3/8/2012 2:38PM</td>
<td>65.87.165.114</td>
</tr>
<tr>
<td>1331244570</td>
<td>3/8/2012 2:09PM</td>
<td>71.10.106.181</td>
</tr>
<tr>
<td>1331261196</td>
<td>3/8/2012 6:46PM</td>
<td>76.102.156.138</td>
</tr>
</tbody>
</table>

Source: Apache Arrow
## Row vs Columnar Layouts

<table>
<thead>
<tr>
<th>Row</th>
<th>session_id</th>
<th>timestamp</th>
<th>source_ip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1331246660</td>
<td>3/8/2012 2:44PM</td>
<td>99.155.155.225</td>
</tr>
<tr>
<td>2</td>
<td>1331246351</td>
<td>3/8/2012 2:38PM</td>
<td>65.87.165.114</td>
</tr>
<tr>
<td>3</td>
<td>1331244570</td>
<td>3/8/2012 2:09PM</td>
<td>71.10.106.181</td>
</tr>
<tr>
<td>4</td>
<td>1331261196</td>
<td>3/8/2012 6:46PM</td>
<td>76.102.156.138</td>
</tr>
</tbody>
</table>

Source: Apache Arrow
Apache Avro

- Row-based layout
- Schemas
  - Embedded (file format) or centralized (message format)
  - Supports versioning and evolution
- Optimal for streaming data (i.e. Apache Kafka), as each message is self-contained
Apache Parquet

- Column-based layout
- Optimized for Hadoop/Spark
- Schema is embedded in the file
- Per-column compression
- Push-down predicates during read
- Column chunking allows skipping I/O
Apache Orc

- Column-based layout
- Optimized for Hadoop/Hive
- Optimized for streaming reads
- Per-column compression
- File-level indices
- Push-down predicates during read
- Column stripes provide parallelism
Apache Arrow

- Column-based layout
- Optimized for in-memory use
- IPC file format
- Dictionary encoding
- Zero-copy reads