Enabling geospatial in big data lakes and databases with LocationTech GeoMesa

ApacheCon@Home 2020

James Hughes

James Hughes

- CCRi's Director of Open Source Programs
- Working in geospatial software on the JVM for the last 8 years
- GeoMesa core committer / product owner
- SFCurve project lead
- JTS committer
- Contributor to GeoTools and GeoServer

Big Geospatial Data

- What type?
- What volume?

Problem Statement for today:

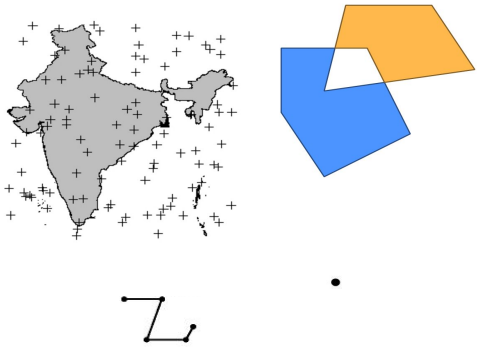
Problem: How do we handle “big” geospatial data?

Problem Statement for today:

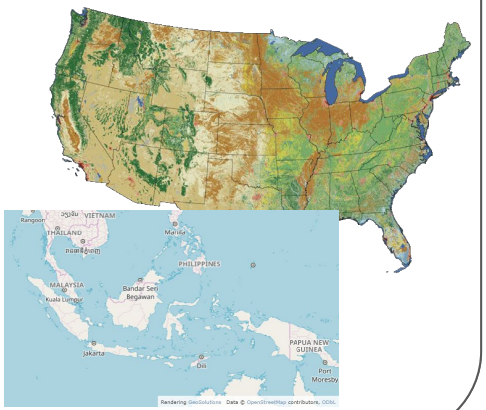
Problem: How do we handle “big” geospatial data?

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

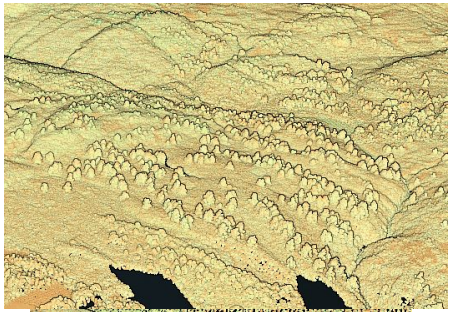
Vector



Raster



Point Cloud

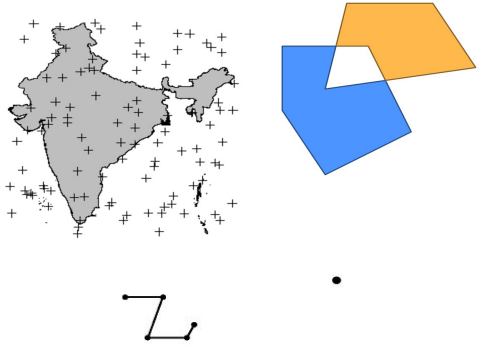


Problem Statement for today:

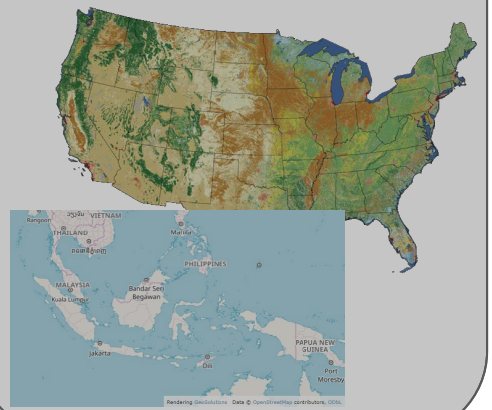
Problem: How do we handle “big” geospatial data?

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

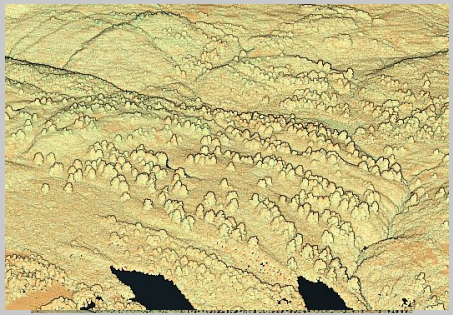
Vector



Raster



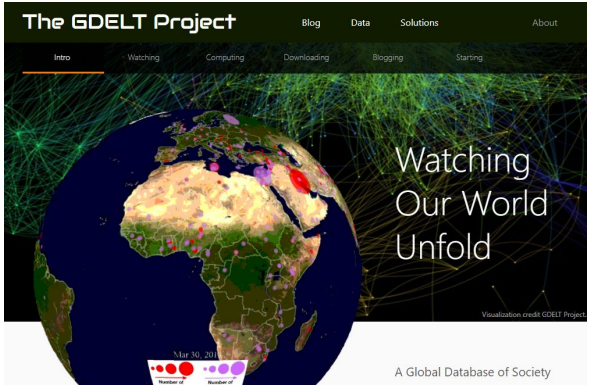
Point Cloud



Problem Statement for today:

Problem: How do we handle “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

Problem Statement for today:

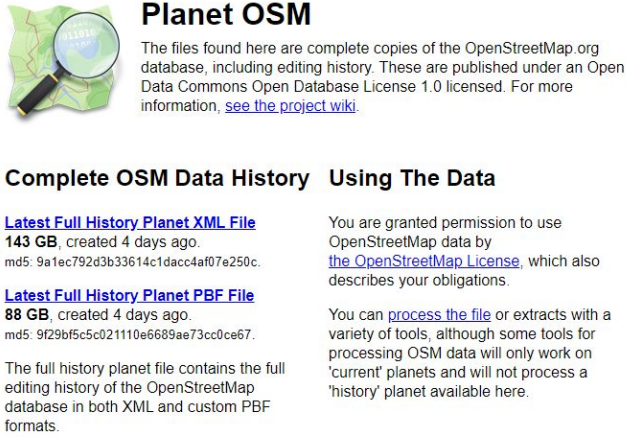
Problem: How do we handle “big” vector geospatial data?

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.



Planet OSM

The files found here are complete copies of the OpenStreetMap.org database, including editing history. These are published under an Open Data Commons Open Database License 1.0 licensed. For more information, see the project wiki.

Complete OSM Data History

Latest Full History Planet XML File
143 GB, created 4 days ago.
md5: 9a1ec792d3b33614c1dacc4af07e250c.

Latest Full History Planet PBF File
88 GB, created 4 days ago.
md5: 9f29bf5c5c021110e6689ae73cc0ce67.

The full history planet file contains the full editing history of the OpenStreetMap database in both XML and custom PBF formats.

Using The Data

You are granted permission to use OpenStreetMap data by the OpenStreetMap License, which also describes your obligations.

You can process the file or extracts with a variety of tools, although some tools for processing OSM data will only work on 'current' planets and will not process a 'history' planet available here.

Problem Statement for today:

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

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

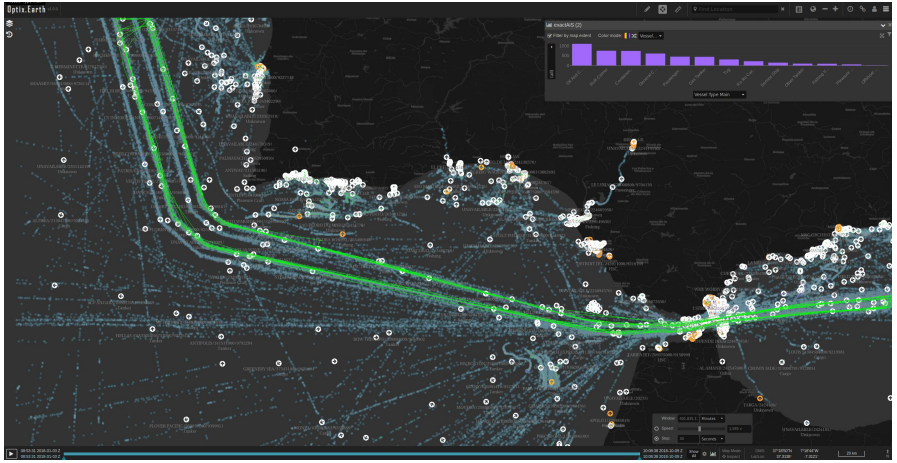
AIS / ADS-B / Mobility data

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!



Problem Statement for today:

Problem: How do we handle millions to billions of vector data (typically points) arriving daily?

LocationTech GeoMesa

CCRi
DATA TO KNOWLEDGE

- GeoMesa Overview
- Reference Architecture

CCRi
DATA TO KNOWLEDGE

What is GeoMesa?

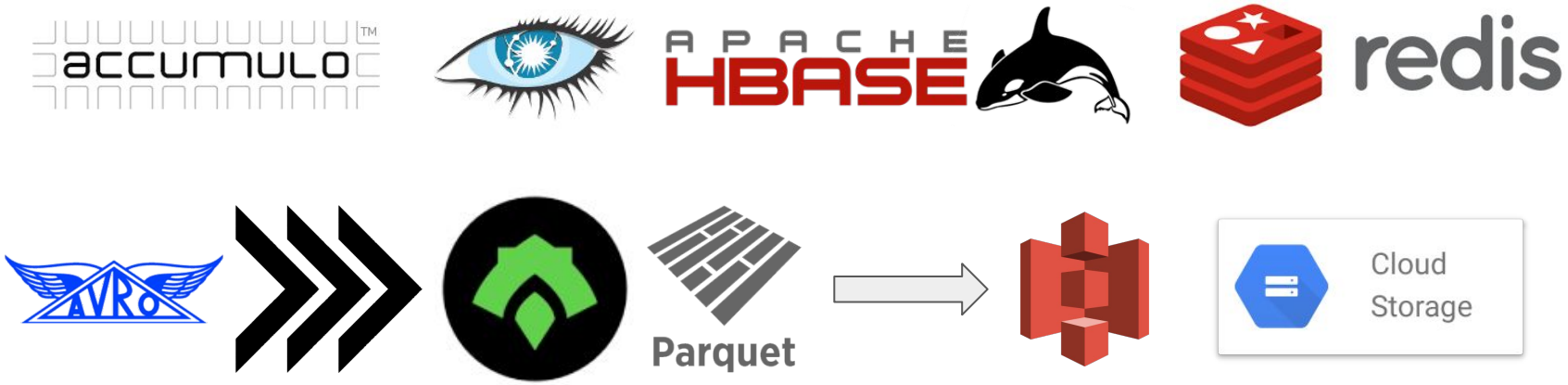
A suite of tools for streaming, persisting, managing, and analyzing spatio-temporal data at scale

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

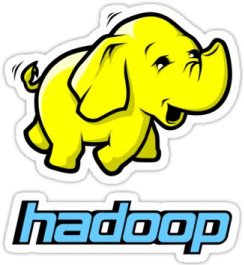


What is GeoMesa?

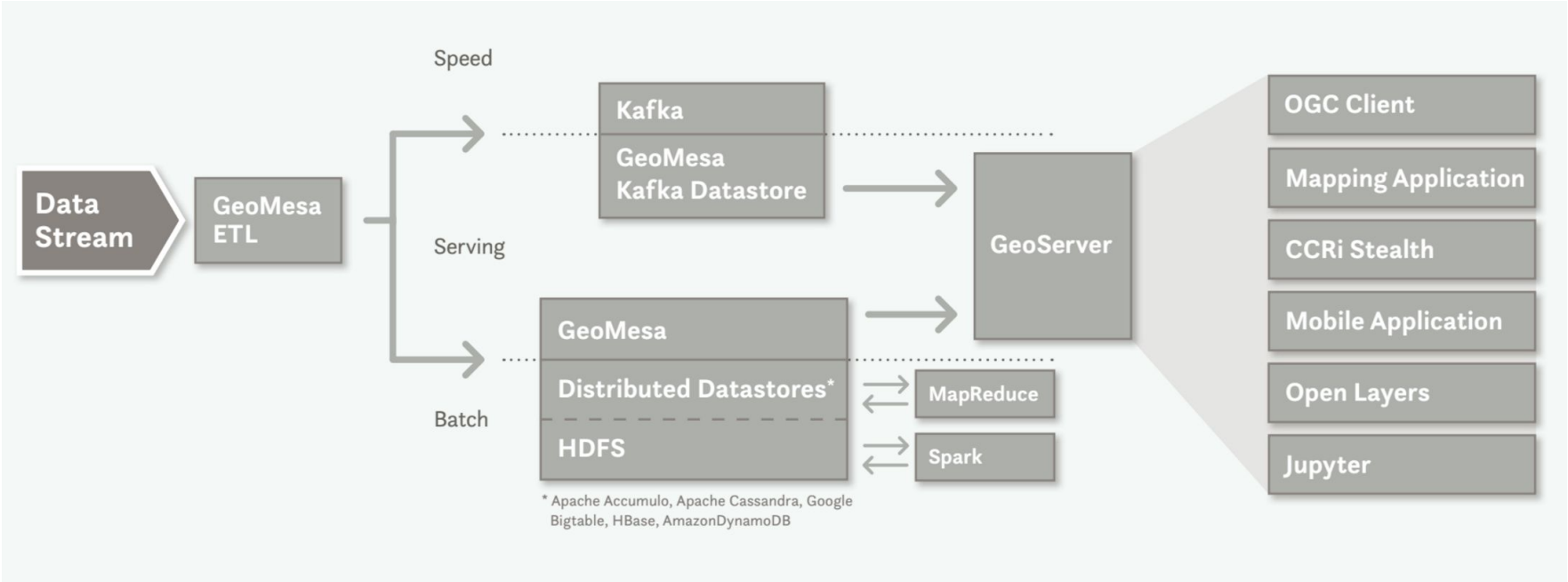
A suite of tools for streaming, persisting, **managing**, and analyzing spatio-temporal data at scale

What is GeoMesa?

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

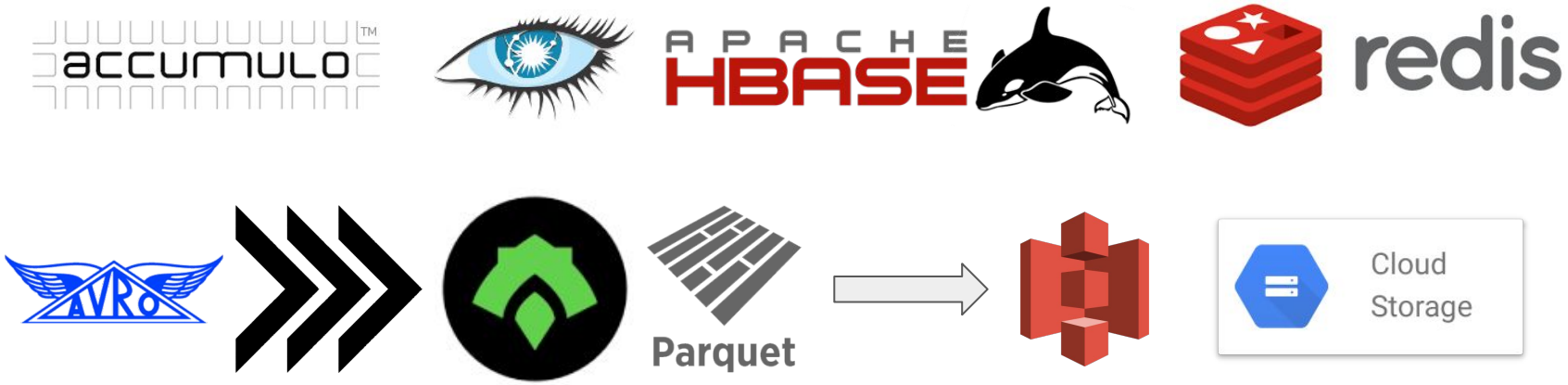


Proposed Reference Architecture



What is GeoMesa?

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



GeoMesa's Persistence

- Distributed Databases
  - Accumulo, HBase, Cassandra
- File formats
  - Arrow, Avro, Orc, Parquet

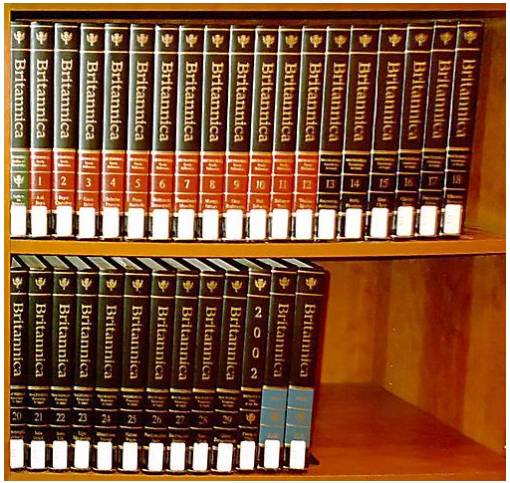
Distributed Key-Value Stores

Accumulo / HBase / Cassandra / Redis

CCRi
DATA TO KNOWLEDGE

Distributed Key-Value Stores

Key-Value Stores provide an ordered view of **keys** with a mapping to some **value**.



accumulo

APACHE HBASE

redis

Indexing Geospatial Data In Key-Value Stores

Accumulo / HBase / Cassandra / Redis

CCRi
DATA TO KNOWLEDGE

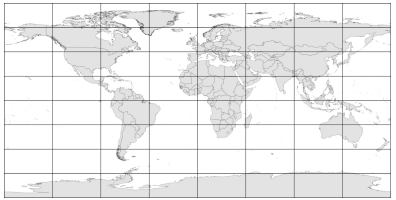
- Key Design using Space Filling Curves

Space Filling Curves (in one slide!)

- **Goal: Index 2+ dimensional data**
- **Approach: Use Space Filling Curves**

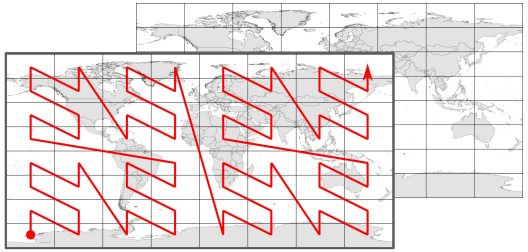
Space Filling Curves (in one slide!)

- **Goal: Index 2+ dimensional data**
- **Approach: Use Space Filling Curves**
- First, ‘grid’ the data space into bins.



Space Filling Curves (in one slide!)

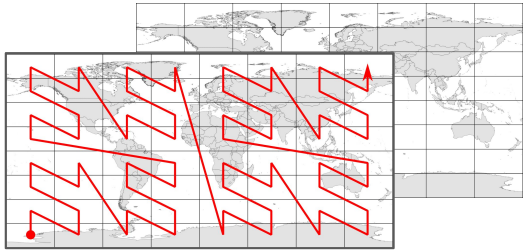
- **Goal: Index 2+ dimensional data**
- **Approach: Use Space Filling Curves**
- First, ‘grid’ the data space into bins.
- Next, order the grid cells with a space filling curve.
    - Label the grid cells by the order that the curve visits the them.
    - Associate the data in that grid cell with a byte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



Z2 "GeoHash"

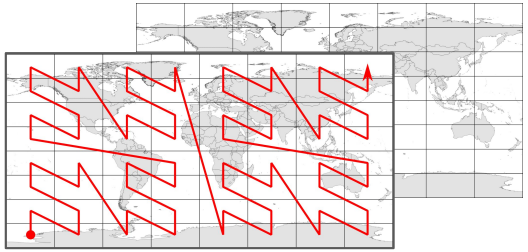
Space Filling Curves (in one slide!)

- **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 the 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.



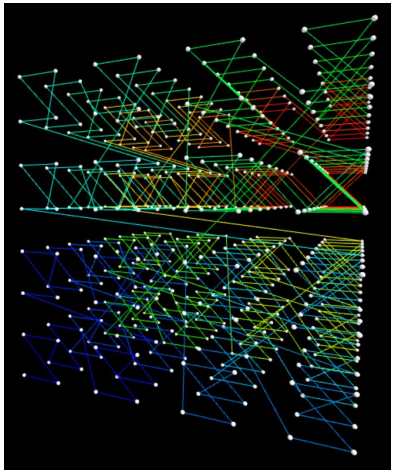
Space Filling Curves (in one slide!)

- **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 the 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.



Z2 "GeoHash"

Z3



Space Filling Curves Applications

This indexing approach can be used to build **bit/byte-based keys** for key-values or build **index strings** for partitioning when using files.

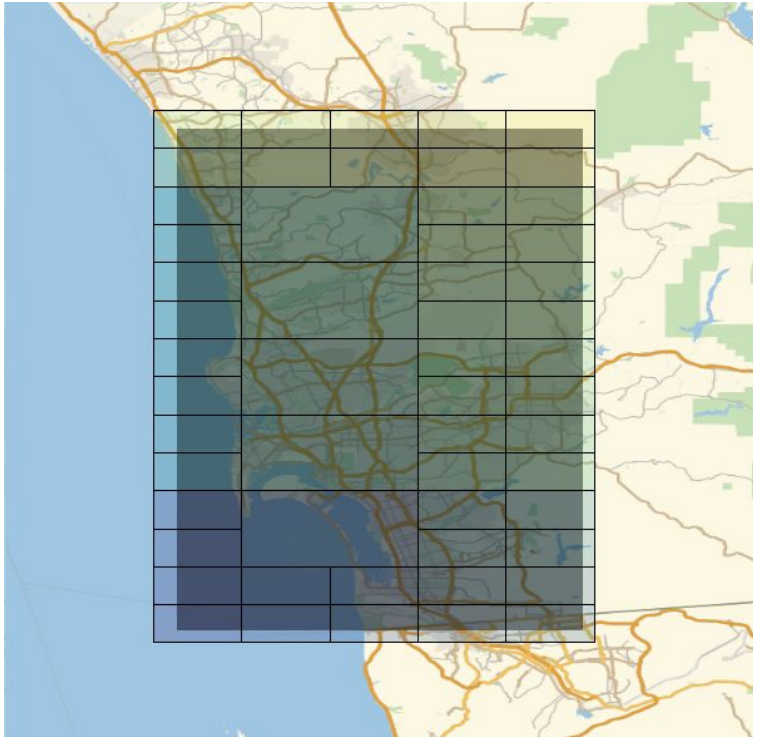
With the indexing approach understood, we can store data in our databases.

Query planning with Space Filling Curves

To query for points in the grey rectangle, the query planner enumerates a collection of index ranges which cover the area.

> Note: Most queries won't line up perfectly with the gridding strategy.

Further filtering can be run on the tablet/region servers (next section)
or we can return 'loose' bounding box results (likely more quickly).



Server-Side Optimizations

Filtering and transforming records in Accumulo and HBase

- Pushing down data filters
  - Z2/Z3 filter
  - CQL Filters
- Projections

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.

GeoMesa Server-Side Filters

- Z2/Z3 filter
  - Scan ranges are not decomposed enough to be very accurate - fast bit-wise comparisons on the row key to filter out-of-bounds data
- CQL/Transform filter
  - If a predicate is not handled by the scan ranges or Z filters,
    then slower GeoTools CQL filters are applied to the serialized SimpleFeature in the row value
  - Relational projections (transforms) applied to reduce the amount of data sent back
- Other specialized filters
  - Age-off for expiring rows based on a SimpleFeature attribute
  - Attribute-key-value for populating a partial SimpleFeature with an attribute value from the row
  - Visibility filter for merging columns into a SimpleFeature when using attribute-level visibilities

Server-Side Optimizations

Aggregations in Accumulo and HBase

- Generating heatmaps
- Descriptive Stats

CCRi
DATA TO KNOWLEDGE

Aggregations

Using Accumulo Iterators and HBase coprocessors, it is possible to aggregate multiple key-value pairs into one response. Effectively, this lets one implement ***map*** and ***reduce*** algorithms.

These aggregations include computing ***heatmaps***, ***stats***, and ***custom data formats***.

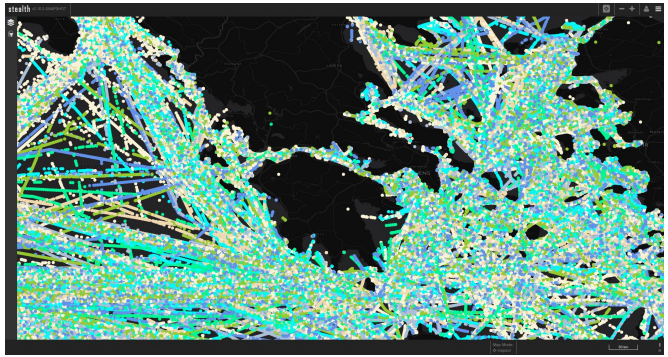
The ability to aggregate data can be composed with filtering and projections.

Notably, all the algorithms we describe work in a single pass over the data.

Visualization Example: Heatmaps

Without powerful visualization options, big data is big nonsense.

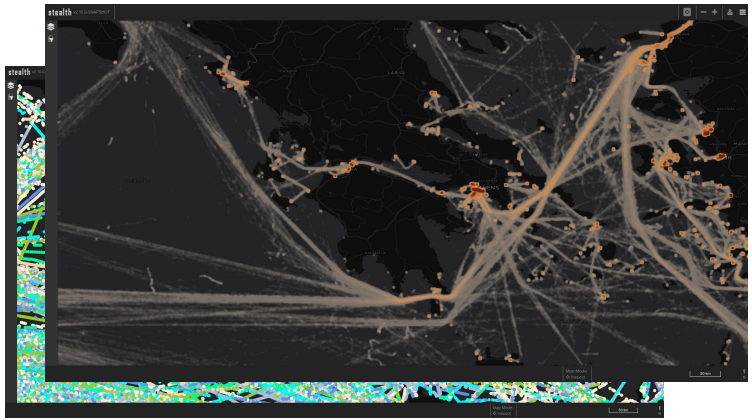
Consider this view of shipping in the Mediterranean sea



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.

Geospatial Databases in Practice

CCRi
DATA TO KNOWLEDGE

Geospatial Databases in Practice

- Accumulo and HBase can both use cloud object storage (AWS S3, Azure, Google equivalents) (reduces costs)
- Various cloud vendors provide ways to run Accumulo and/or HBase more easily

GeoMesa 3.x supports

- Accumulo 1.x and 2.0
- HBase 1.4.x and 2.2.x

Optimizing Big Data Formats for Vector Data

CCRi
DATA TO KNOWLEDGE

- File formats overview
- Spatial extensions to file formats

Specialized Big data file formats

Benefits of big data file formats

- Columnar layouts
- Dictionary encoding
- Efficient compression
- Structured
- Optimized filtering on read
- Language interoperability

Benefits of big data file formats

- Columnar layouts
- Dictionary encoding
- Efficient compression
- Structured
- Optimized filtering on read
- Language interoperability

One problem!

- No spatial types!

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

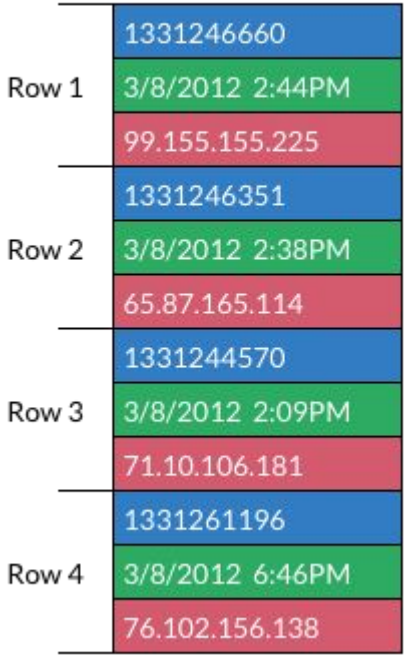
Row vs Columnar Layouts

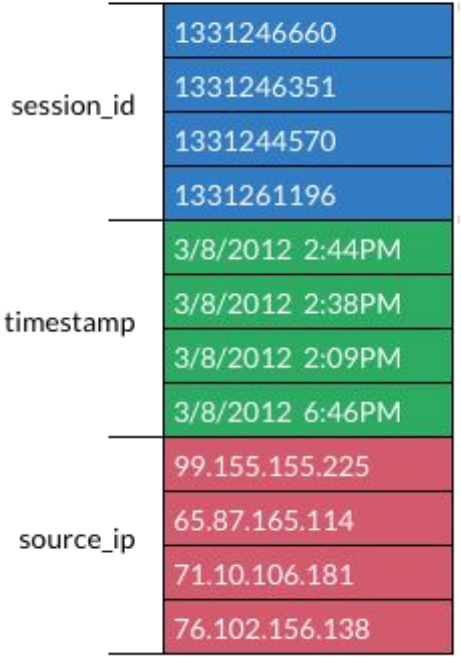
	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138

Source: Apache Arrow

Row vs Columnar Layouts

	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138





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

CCRi
DATA TO KNOWLEDGE

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

Spatial File Formats in GeoMesa

- No native spatial types
- Geometries are built up with lists of primitive columns
- Similar to GeoJSON, can be read without special type awareness

Spatial File Formats in GeoMesa

- Points
  - Stored as two columns of type Double, one for X and one for Y
  - Arrow - stored as tuples (FixedSizeList)
- Allows for push-down filtering against each dimension

Spatial File Formats in GeoMesa

- Points
  - Stored as two columns of type Double, one for X and one for Y
  - Arrow - stored as tuples (FixedSizeList)
- Allows for push-down filtering against each dimension
- LineStrings, MultiPoints
  - Stored as two columns of type List[Double]

Spatial File Formats in GeoMesa

- Points
  - Stored as two columns of type Double, one for X and one for Y
  - Arrow - stored as tuples (FixedSizeList)
- 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

Spatial File Formats in GeoMesa

- Points
  - Stored as two columns of type Double, one for X and one for Y
  - Arrow - stored as tuples (FixedSizeList)
- 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
- Avro - row based - WKB/TWKB/WKT

Reading and Writing Spatial Formats

- Parquet
  - geomesa-fs-storage-parquet - SimpleFeatureParquetWriter, FilteringReader
- Orc
  - geomesa-fs-storage-orc - OrcFileSystemReader/Writer
- Arrow
  - geomesa-arrow-jts - PointVector, LineStringVector, etc
- Avro
  - geomesa-feature-avro - AvroFeatureSerializer, AvroDataFileReader/Writer

Reading and Writing Spatial Formats

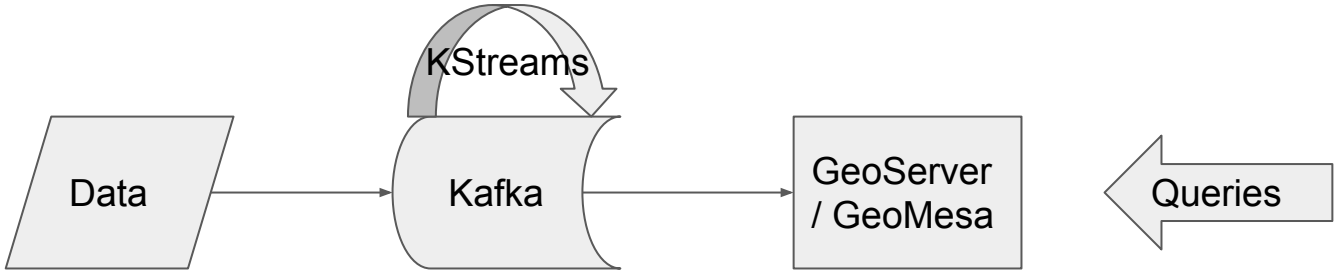
- Parquet/Orc
  - GeoMesa file system data store
  - GeoMesa CLI export/ingest
- Arrow
  - WFS/WPS requests through GeoServer
  - GeoMesa CLI export
- Avro
  - WFS/WPS requests through GeoServer
  - GeoMesa CLI export/ingest
- Standard format tools

Spatial File Format Use Cases

- Streaming Data
- Spark analytics
- ETL and tiered storage
- Data visualization

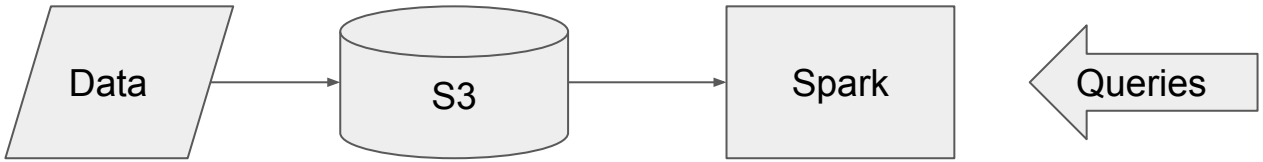
Streaming Data - Apache Avro

- Each message is a single record (row based)
- Apache Kafka/Streams for data exchange
- Confluent schema registry is used for managing schemas
  - Small header per message uniquely identifies schema
  - Schema evolution for adding/removing fields
- GeoMesa Kafka data store for in-memory indexing



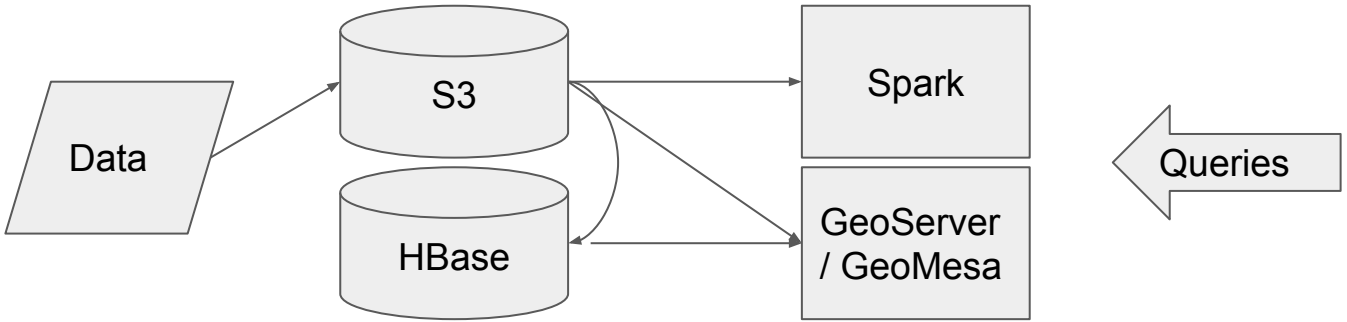
Spark Analytics - Apache Parquet and Orc

- GeoMesa Spark integration adds spatial UDFs/UDTs
  - st_contains, st_point, etc
- Native input formats provide high throughput
- Relational projections take advantage of columnar layouts
- Predicates are pushed down into the file reads



Tiered Storage and ETL - Apache Parquet and Orc

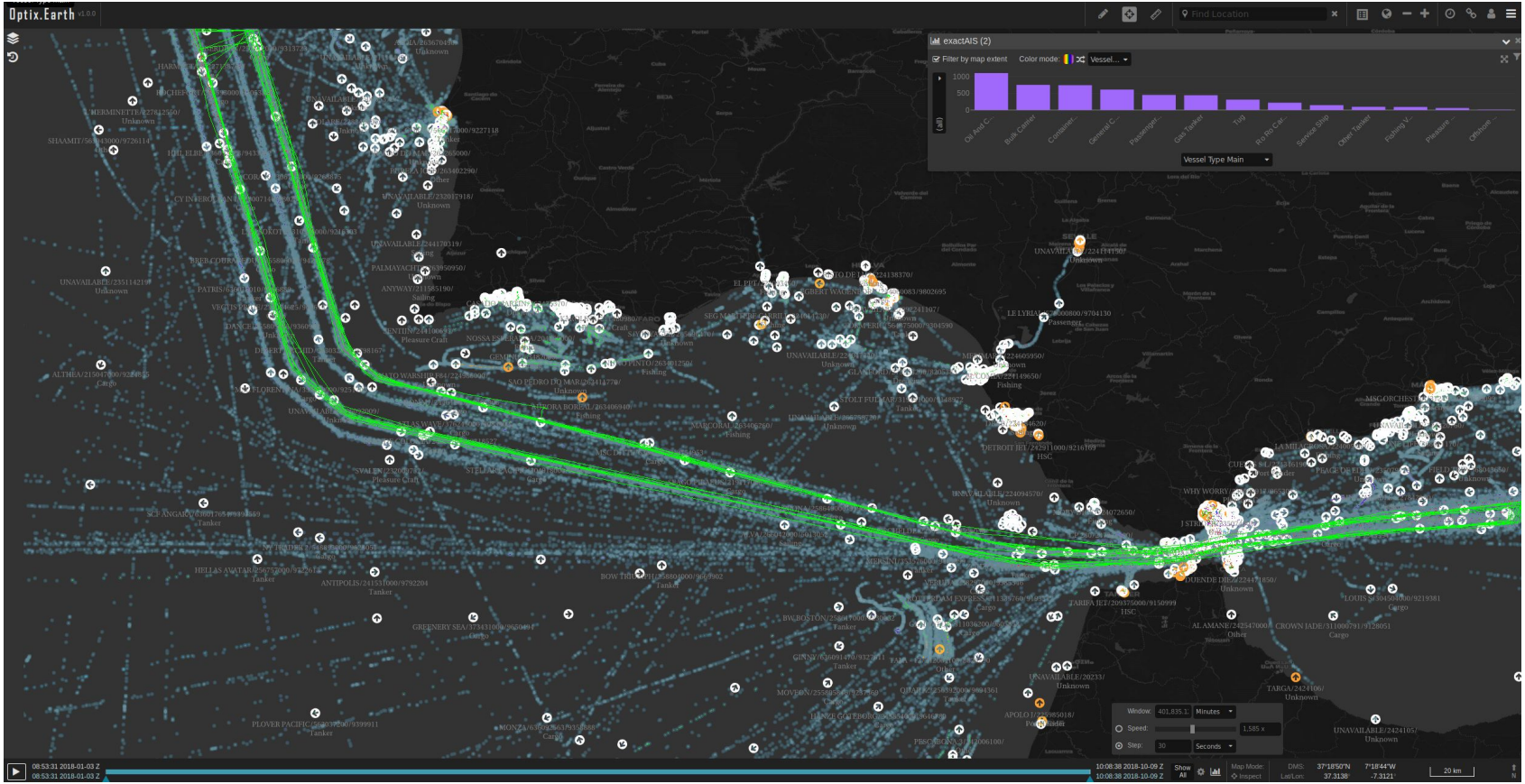
- Data is pre-processed into S3 using the GeoMesa converter library to create Parquet or Orc
- Processed files are ingested directly from S3 into HBase
- Processed files are accessed with the GeoMesa file system data store for large-scale analytics
- Data age-off is used to keep your HBase cluster small
- Merged view data store shows combined HBase + S3



Data Visualization - Apache Arrow

- Query Arrow IPC data through WFS/WPS
  - Distributed aggregation used where possible
- Arrow-js wraps the raw bytes and exposes the underlying data
- Can efficiently filter, sort, count, etc to display maps, histograms, timelapses

Data Visualization in browser with Apache Arrow



Geospatial File formats in Practice

CCRi
DATA TO KNOWLEDGE

Geospatial File formats in Practice

Avro is great for interchange between systems

Arrow is great for analysis use cases

Orc/Parquet have benefits for long-term, data lake storage use cases

- Space-filling curves can be used to generate partition strings
- File and block level information in Orc/Parquet can help with very coarse spatial filtering
- Sorting inside a file can provide additional compression benefits

Thanks!

James Hughes

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

CCRi is hiring!
https://www.ccri.com/careers/

- DevOps
- Software Engineers
- Data Scientists