GeoMesa as a Distributed Spatio-Temporal Database and Computational Framework

James Hughes – james.hughes@ccri.com
What is GeoMesa?

- Distributed Spatio-Temporal Database
  - Built on Hadoop and Accumulo
- OGC Standards-based Data Access
  - GeoTools & GeoServer
- LocationTech Open Source
  - Eclipse Foundation
- Written in Scala;}


Database vs. Computational Framework

- A database provides an API for entering a query and getting a result set.
- A computational framework provides access to APIs which allow for some computation to be applied to the result from a query.
- Database: Query => Result Set
- CF: Query, function => function(Result Set)
Part 1: GeoMesa as a ST Database

- Use case (voluminous spatio-temporal data)
- Open source, Open standards, Big Data
- Accumulo
- GeoMesa Index: GeoHashes on Accumulo
Part 2: GeoMesa Computation

- Three GeoMesa Example Analytics
- Collaborations
  - A Walk with a co-worker
  - Fall and Spring FOA UT-Austin Projects
  - SFCurve
- View forward – Spark, Raster Support, etc.
Example Dataset: GDELT

Global Data on Events, Language, and Tone

213M Geolocated Events since 1979
GeoMesa Use Case

- GeoMesa is for when:
  - Traditional RDBMs will not scale suitably.
  - A cloud/cluster is available.
  - Distributed storage and distribute processing is desired.
  - A F/OSS option is preferred.
Scaling up...

• How do we store increasing amounts of data?
  – Commercial options exist to scale up with proprietary, etc. (Expensive)
  – Application-level sharding. (Custom)
  – Give up, only batch process the data. (No online access, long jobs.)
  – Consider cloud/non-traditional approaches.
Hadoop Infrastructure

Cloud Stack and Distributed Resources

- MapReduce
- Spark
- Storm
- GeoMesa
- Accumulo
- HDFS

Certified OGC Compliant
Accumulo

- Accumulo is a BigTable clone which adds server-side computation via Iterators as well as cell-level security.
- Accumulo uses HDFS as its persistence layer.
- Accumulo is a (sorted) key-value store with keys like:
Apache Accumulo

- Tables composed of Tablets
- Tablet Servers
- Partitioning Data
- Parallel Queries

### Data Distribution

<table>
<thead>
<tr>
<th>Table</th>
<th>Partitioning</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>rowA</td>
<td>col2 1</td>
<td></td>
</tr>
<tr>
<td>rowB</td>
<td>col1 4</td>
<td></td>
</tr>
<tr>
<td>rowC</td>
<td>col1 1</td>
<td>rowB</td>
</tr>
<tr>
<td>rowC</td>
<td>col2 2</td>
<td>rowB</td>
</tr>
<tr>
<td>rowC</td>
<td>col3 6</td>
<td>rowB</td>
</tr>
<tr>
<td>rowF</td>
<td>col2 4</td>
<td>rowA</td>
</tr>
<tr>
<td>rowH</td>
<td>col2 5</td>
<td>rowA</td>
</tr>
<tr>
<td>rowH</td>
<td>col3 8</td>
<td>rowA</td>
</tr>
</tbody>
</table>
Accumulo Simply

- (Get) Binary structures allows fast look-up.
- (Scan) Linked list structure allows for sequential reads.
- Think “Multi-volume encyclopedia.”
Indexing Multi-dimensional data

- Accumulo orders data in 1d.
- Spatio-temporal data is at least 3d.
- Desired properties:
  - Nesting – a substring of a an address should reference a containing rectangle.
  - Locality – addresses lexical similar should be close together.
GeoHash/Z-Order Curve
Part 2: GeoMesa Computation

- Three GeoMesa Example Analytics
- Collaborations
  - A Walk with a co-worker
  - Fall and Spring FOA UT-Austin Projects
  - SFCurve
- View forward – Spark, Raster Support, etc.
Ex 1: Parallel Density Calculations
Ex 1: Parallel Density Calculations
Accumulo Iterator allows us to perform server-side 'Map' and 'Reduce' computations.

We then have to 'Reduce' the results again on the client.
Ex 2: DBSCAN Clustering

- **Density-based Spatial Clustering of Applications with Noise**
- Clusters Algorithmically
  - No need to specify $k$ (number of clusters)
- Arbitrarily shaped clusters
Tweet Clustering - St. Louis
Ex 3: Using Spark to build a Time-Series

- Apache Spark
  - Real-time data analysis
  - RDD (Resilient Distributed Dataset)
    - Transformations
    - Filters
    - Aggregations
  - Example Time Series Output
    - Visualize via R
/ Ex: 3 Cont.
Getting a RDD[SimpleFeature]

// Get a handle to the data store
val params = Map(
  "instanceId" -> "myinstance",
  "zookeepers" -> "zool,zoo2,zoo3",
  "user" -> "username",
  "password" -> "password",
  "tableName" -> "geomesa_catalog"
)

val ds = DataStoreFinder.getDataStore(params).asInstanceOf[AccumuloDataStore]

// Construct a CQL query to filter by bounding box
val ff = CommonFactoryFinder.getFilterFactory2
val f = ff.bbox("geom", -90.32023,38.72009,-90.23957,38.77019, "EPSG:4326")
val q = new Query(feature, f)

val conf = new Configuration
val sconf = init(new SparkConf(true), ds)
val sc = new SparkContext(sconf)

val queryRDD = geomesa.compute.spark.GeoMesaSpark.rdd(conf, sconf, ds, query)
Ex: 3 Cont.
Computing the Time-Series

// Convert RDD[SimpleFeature] to RDD[(String, SimpleFeature)] where the first
// element of the tuple is the date to the day resolution
val dayAndFeature = queryRDD.mapPartitions { iter =>
  val df = new SimpleDateFormat("yyyyMMdd")
  val ff = CommonFactoryFinder.getFilterFactory2
  val exp = ff.property("dtg")
  iter.map { f => (df.format(exp.evaluate(f).asInstanceOf[java.util.Date]), f) }
}

// Aggregate and output
val groupedByDay = dayAndFeature.groupBy { case (date, _) => date }
val countByDay = groupedByDay.map { case (date, iter) => (date, iter.size) }
countByDay.collect.foreach(println)
Ex: 3 Finished
R Visualization
OpenSource Geospatial Analytic Pipeline
A Walk with Tim
A Walk with Tim

- After chatting with Tim, I realized GeoMesa could use
  - A Temporal Density Iterator
  - An Iterator for Tim's project (aka Mantis iterator)
  - Other examples of iterators which could work in the 'Map/Reduce' and 'Reduce' framework. (Effectively anything with associative aggregators.)
Fall 2014 FOA Students

http://skcooper-cs370.blogspot.com/
http://cs370fborbrooks.blogspot.com/
https://timsherlock370.wordpress.com/
The students implemented the TemporalDensityIterator
- Implemented Unit Tests.
- Wired it up as a WPS process.
- Worked toward connecting it to a basic UI.

```scala
import org.locationtech.geomesa.utils.geotools.SimpleFeatureTypes, Time
import org.opengis.feature.simple.SimpleFeatureType
import scala.util.Random

class TemporalDensityIterator(other: TemporalDensityIterator, env: Iterator)
```
Spring 2015 FOA GM Projects

• Build up modular UI to demo existing GeoMesa analytics such as the Temporal Density Iterator.

• Explore GeoHash, Hilbert curve, and SFC implementations.
SF Curve Project

SF Curve

Basics

This proposal is in the Project Proposal Phase (as defined in the Eclipse Development Process) and is written to declare its intent and scope. We solicit additional participation and input from the community. Please login and add your feedback in the comments section.

Parent Project:
Technology

Background:
The SF Curve library was conceived as part of a collaborative effort between two LocationTech projects, GeoTrellis and GeoMesa. Each project had developed Scala code to facilitate the creation and manipulation of Z-order curve indices based on spatial or spatiotemporal properties of data. In GeoTrellis, the Z-order curve functionality allows for raster tiles to be indexed such that the execution of large range queries and distributed raster operations is optimized. GeoMesa has analogous code to construct a GeoHash that serves a similar purpose for processing vector data.

The GeoTrellis and GeoMesa teams recognized that the similar code they had developed in each independent project was essentially solving the same problems, and the decision was made to collaborate on a common library for solving this set of problems. SF Curve is that library.
New: GeoMesa Raster Support

- GeoMesa has completed a project to implement raster support.
- We're looking forward to collaborating with other LocationTech projects on this effort.
Distributed Raster Computation

Model 1: Pixel by Pixel => Tile by Tile => Row by Row

Example 1: ‘Map’ Task:

Convert a collection of color tiles to grayscale.
Distributed Raster Computation

Model 1: Pixel by Pixel => Tile by Tile => Row by Row

Example 2: ‘Map/Reduce’ Task:
Calculate a histogram.
Distributed Raster Computation

Model 2: ‘Reduce/Collect’ by GeoHash

Example 1: ‘Reduce’ Task:

Build Image Pyramid
Conclusion

- GeoMesa indexes big geo-time data.
- A simple computational framework is available via server side programming.
- More complex computation can be approached via MapReduce and Spark.
- Future work will continue to focus on scaling user interactions.
Questions?

Tutorials, Presentations, Paper, Code, & more

http://geomesa.org

http://github.com/locationtech/geomesa

Email

geomesa-users@locationtech.org