# Optimizing Big Data Formats for Vector Data

Emilio Lahr-Vivaz

FOSS4G-NA, April 17, 2019


@CCR_inc

# Agenda

1. Introduction
2. Row vs column layouts
3. Overview of big data file formats
4. Spatial extensions to file formats
5. Use case: Streaming data
6. Use case: Spark analytics
7. Use case: ETL and tiered storage
8. Use case: Data visualization

# Introduction

- Big data requires specialized file formats
  - ETL, messaging, archiving, visualizations, storage costs

# Introduction

- Big data requires specialized file formats




# Introduction

- Big data requires specialized file formats
- Benefits
  - Columnar layouts
  - Dictionary encoding
  - Efficient compression
  - Structured
  - Optimized filtering on read
  - Language interoperability

# Introduction

- Big data requires specialized file formats
- Benefits
  - Columnar layouts
  - Dictionary encoding
  - Efficient compression
  - Structured
  - Optimized filtering on read
  - Language interoperability
- 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

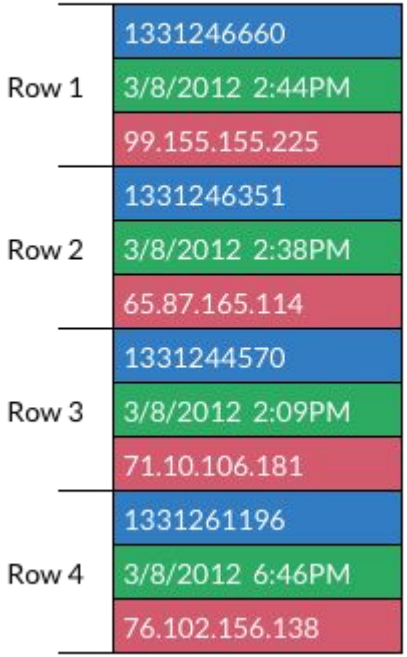
# Row vs Columnar Layouts

| | 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

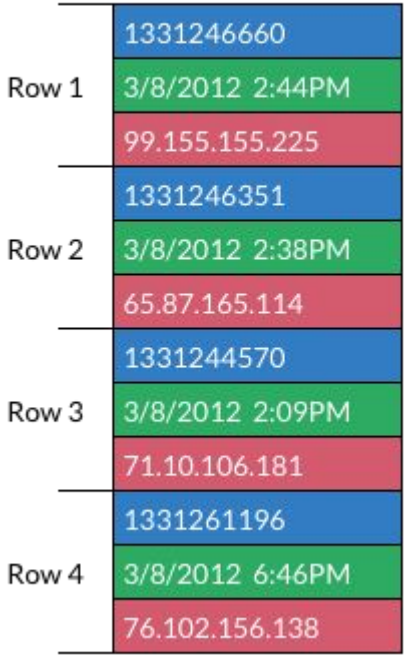
| | session_id | timestamp | source_ip |
|---|---|---|---|
| Row 1 | 1331246660 | 3/8/2012 2:44PM | 99.155.155.225 |
| Row 2 | 1331246351 | 3/8/2012 2:38PM | 65.87.165.114 |
| Row 3 | 1331244570 | 3/8/2012 2:09PM | 71.10.106.181 |
| Row 4 | 1331261196 | 3/8/2012 6:46PM | 76.102.156.138 |

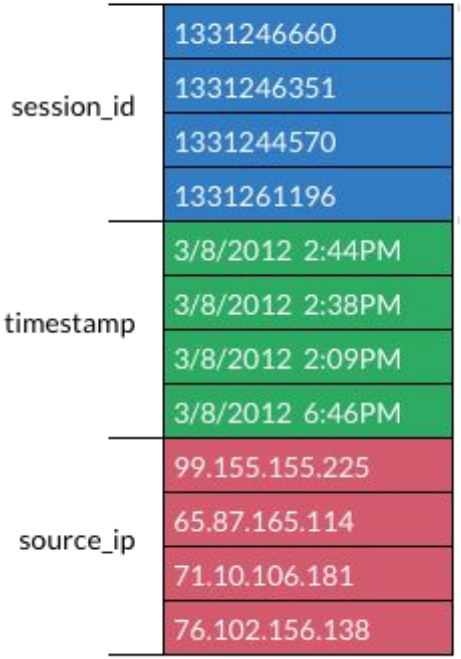



Source: Apache Arrow

# 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

# 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

# Overview of GeoMesa

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

# Overview of GeoMesa

- GeoTools data store implementations for HBase, Accumulo, Cassandra, Bigtable, Redis, Kafka, **S3**, etc

# Overview of GeoMesa

- GeoTools data store implementations for HBase, Accumulo, Cassandra, Bigtable, Redis, Kafka, **S3**, etc
- Spark spatial UDFs/UDTs and query integration

# Overview of GeoMesa

- GeoTools data store implementations for HBase, Accumulo, Cassandra, Bigtable, Redis, Kafka, **S3**, etc
- Spark spatial UDFs/UDTs and query integration
- Declarative converter library for ETL

# 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

# 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

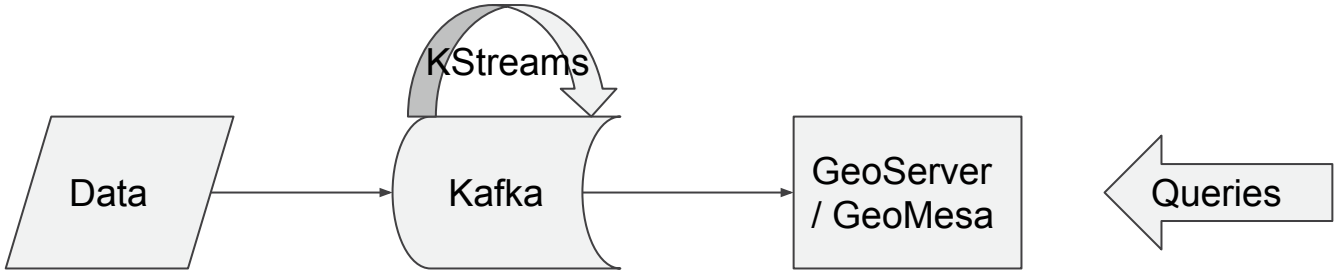
- Parquet
  - geomesa-fs-storage-parquet - SimpleFeatureParquetWriter, FilteringReader
- Orc
  - geomesa-fs-storage-orc - OrcFileSystemReader/Writer
- Arrow
  - geomesa-arrow-jts - PointVector, LineStringVector, etc
- Avro
  - geomesa-feature-avro - AvroFeatureSerializer, AvroDataFileReader/Writer

# 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

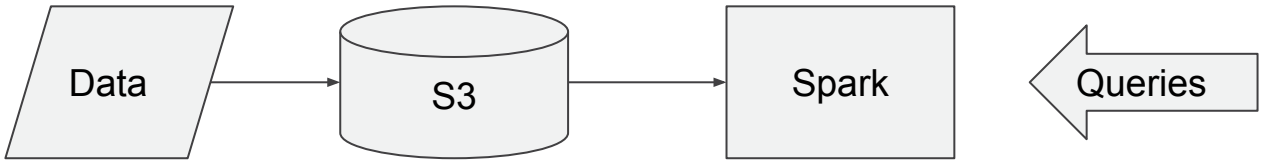
# Streaming Data - Apache Avro

- 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



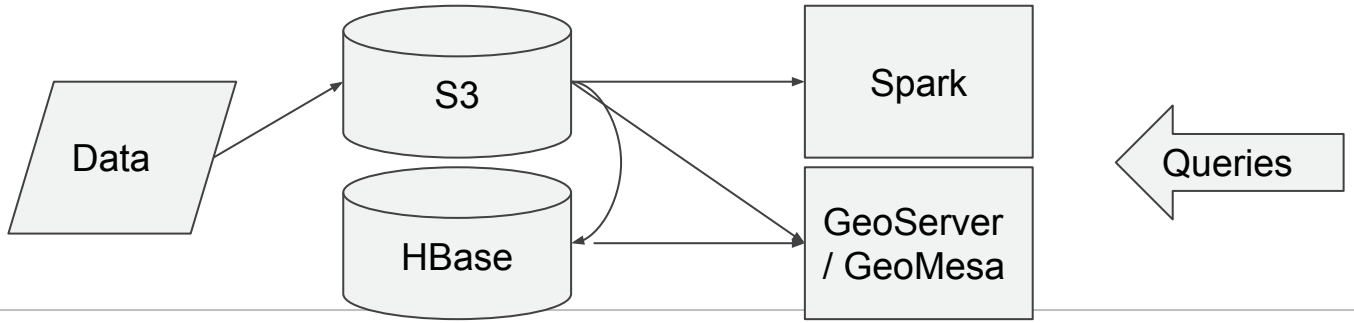
# Spark Analytics - Apache Parquet and Orc

- 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

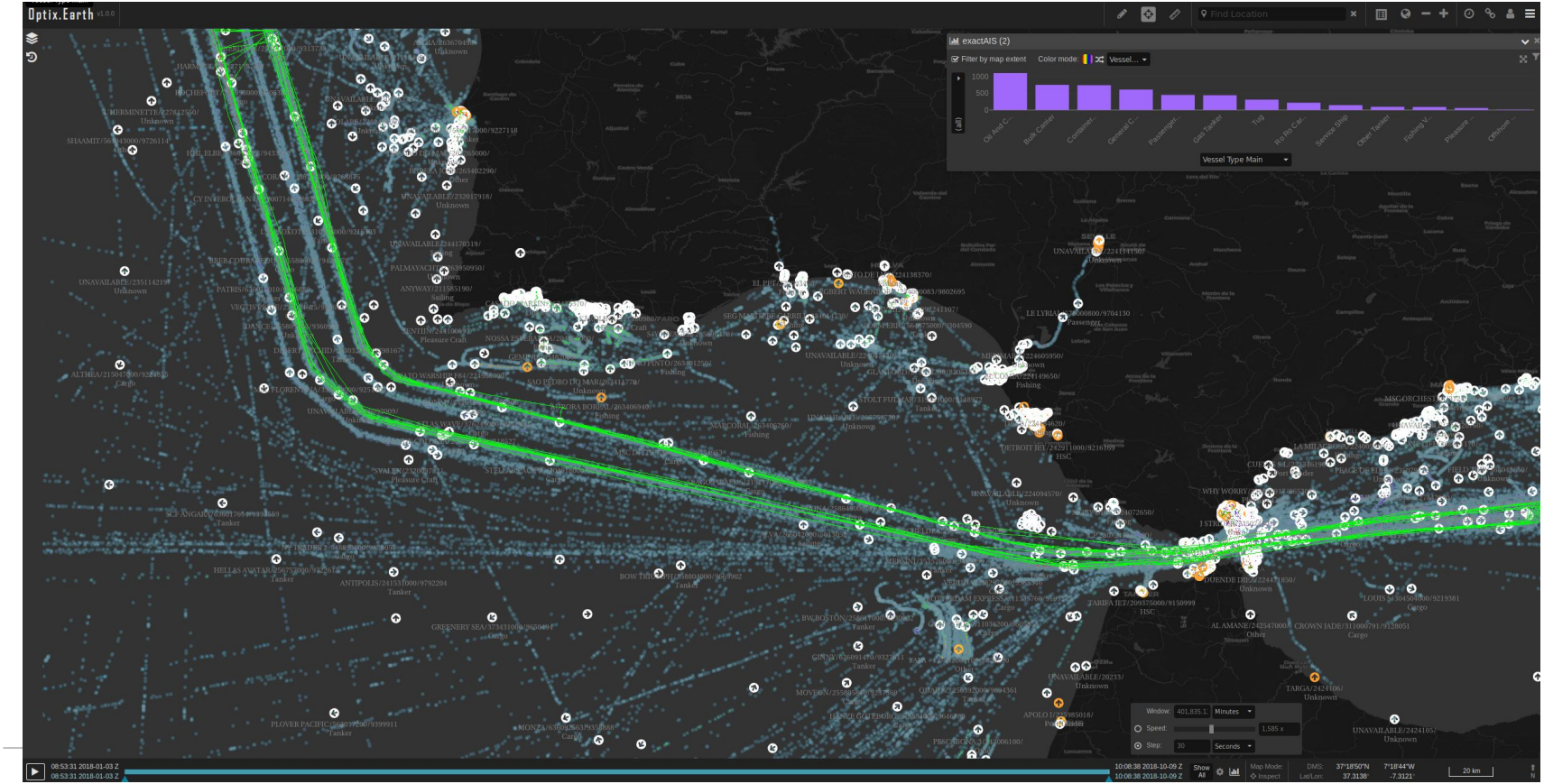
- Data is pre-processed into S3 using the GeoMesa converter library to create Parquet or Orc
- Processed files are ingested directly from S3 into HBase
- Processed files are accessed with the GeoMesa file system data store for large-scale analytics
- Data age-off is used to keep your HBase cluster small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



# Thank you

- geomesa.org
- github.com/locationtech/geomesa
- gitter.im/locationtech/geomesa
- ccri.com
- @CCR_inc