v1.0.0

GeoWave User Guide

Introduction

What is GeoWave

GeoWave is an open-source library for storage, index, and search of multi-dimensional data on top of sorted key-value datastores and popular big data frameworks. GeoWave includes specific tailored implementations that have advanced support for OGC spatial types (up to 3 dimensions), and both bounded and unbounded temporal values. Both single and ranged values are supported on all axes. GeoWave's geospatial support is built on top of the GeoTools project extensibility model. This means that it can integrate natively with any GeoTools-compatible project, such as GeoServer and UDig, and can ingest GeoTools compatible data sources.

GeoWave provides out-of-the-box support for distributed key/value stores, as necessary for mission needs. The latest version of GeoWave supports Apache Accumulo and Apache HBase stores, though additional data stores can be implemented as requested or needed.

This guide serves the purpose of focusing on the development side of GeoWave capabilities as well as assisting developers with the GeoWave code surroundings.

- GeoWave Capabilities
  - Add multi-dimensional indexing capability to Apache Accumulo and Apache HBase
  - Add support for geographic objects and geospatial operators to Apache Accumulo and Apache HBase
  - Provide a GeoServer plugin to allow geospatial data in Accumulo and HBase to be shared and visualized via OGC standard services
  - Provide Map-Reduce input and output formats for distributed processing and analysis of geospatial data
- Geospatial software plugins include the following:
  - GeoServer plugin to allow geospatial data in Accumulo to be shared and visualized via OGC standard services
  - PDAL plugin for working with point cloud data
  - Mapnik plugin for generating map tiles and generally making good looking maps.

Basically, GeoWave is working to bridge geospatial software with distributed compute systems and attempting to do for distributed key/value stores what PostGIS does for PostgreSQL.

Origin

GeoWave was developed at the National Geospatial-Intelligence Agency (NGA) in collaboration with RadiantBlue Technologies and Booz Allen Hamilton. The government has unlimited rights and is releasing this software to increase the impact of government investments by providing developers with the opportunity to take things in new directions. The software use, modification, and distribution rights are stipulated within the Apache 2.0 license.

Intent

Pluggable Backend

GeoWave is intended to be a multidimensional indexing layer that can be added on top of any sorted key-value store. Accumulo was chosen as the initial target architecture, and support for HBase has been added as well. Any datastore which allows prefix based range scans should be straightforward extensions.

Modular Framework Design

The GeoWave architecture is designed to be extremely extensible with most of the functionality units defined by interfaces, and with default implementations of these interfaces to cover most use cases. GeoWave allows for easy feature extension and platform integration – bridging the gap between distributed technologies and minimizing the learning curve for developers. The intent is that the out of the box functionality should satisfy 90% of use cases, but the modular architecture allows for easy feature extension as well as integration into other platforms.

Self-Describing Data

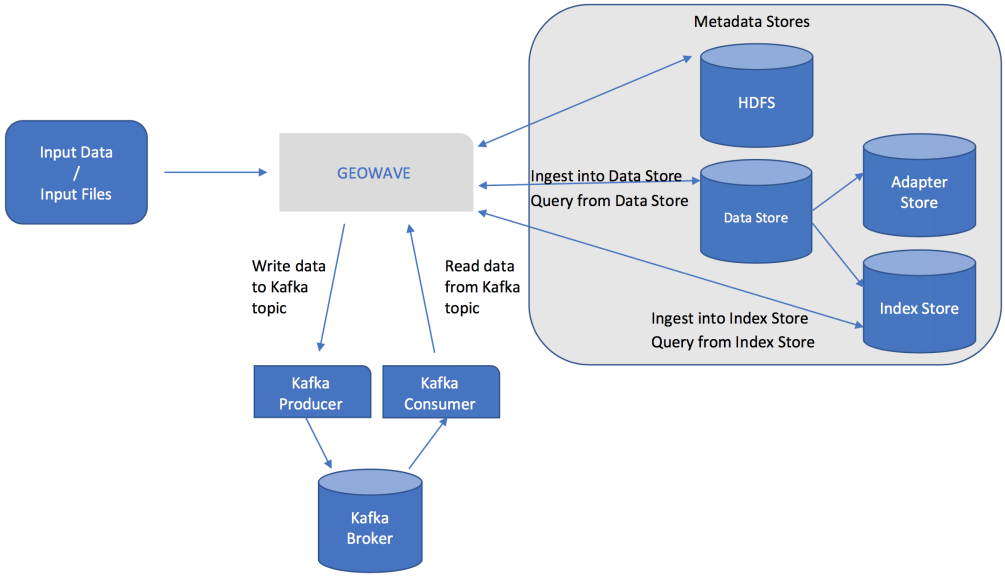
GeoWave stores the information needed to manipulate data, such as configuration and format, in the database itself. This allows software to programmatically interrogate all the data stored in a single or set of GeoWave instances without needing bits of configuration from clients, application servers, or other external stores.

Scalable

GeoWave is designed to operate either in a single-node setup or it can scale out as large as needed to support the amount of data and/or processing resources necessary. By utilizing distributed computing clusters and server-side fine grain filtering, GeoWave is fully capable of performing interactive time and/or location specific queries on datasets containing billions of features with 100 percent accuracy.

Overview

For GeoWave users, the primary method of interfacing with GeoWave is through the various Command-Line Interface (CLI) commands and options. Users will use GeoWave to store, index, or search multi-dimensional data in a key-value datastore.



This *typically* involves these four steps:

- **Configure**

  Set up/configure a datastore or index on GeoWave for re-use across various operations as needed.

- **Ingest/Index**

  Ingest, or Index, data into a specific store (e.g., Accumulo, HBase)

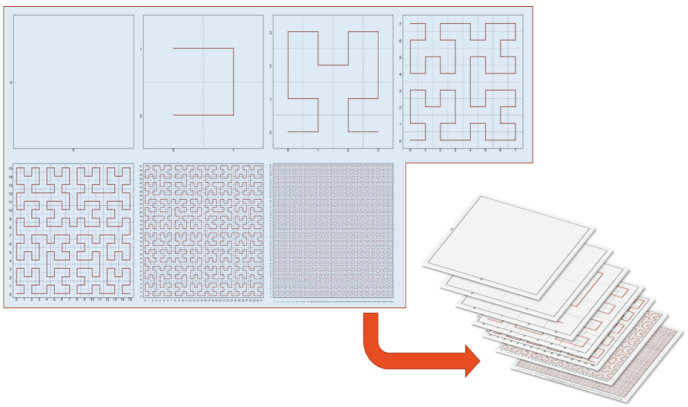
- **Process**

  Process data using a distributed processing engine (e.g. MapReduce, Spark)

- **Query/Discover**

  Search/Query or Discover data ingested, indexed, or processed/transformed through GeoWave operations. A common data discovery tool used is GeoServer, which interfaces with GeoWave through the plugin, for interfacing with the selected datastore, e.g., Accumulo or HBase.

GeoWave uses tiered, gridded, Space Filling Curves (SFCs) to index data into your desired key-value store. The indexing information is stored in a generic key structure that can also be used for server-side processing. This architecture allows query, processing, and rendering times to be reduced by multiple orders of magnitude.



If there are questions or issues encountered, or topics of interest that could be expounded on, please create an issue within the GeoWave project GitHub page.

Assumptions

Prior to running GeoWave functionality, this guide assumes that the following components are installed and/or available. Because of the continuous changes occurring to these components, installing and maintaining extensive operational capabilities around these components is outside the scope of this document.

Running GeoWave

In order to build and/or perform development using the GeoWave source, the following components are required:

- Java Development Kit (JDK) (>= 1.8)

  Requires JRE v1.8 or greater

  Download the latest JRE from the Java downloads site. The OracleJDK is the most thoroughly tested but there are no known issues with OpenJDK.

- GeoWave build or RPM

  At minimum, an existing GeoWave build or RPM is required. For building the GeoWave source, please refer to the development guide.

External Components

Depending on the environment components being developed towards (e.g., requirements, datastores, indices, etc), the following are not all required, though constraints are listed below for which versions are supported by GeoWave.

- GeoServer instance >= 2.12.1
- Apache Accumulo version 1.5 or greater is required. 1.5.x, 1.6.x, 1.7.x, 1.8.x, have all been tested.
- Apache HBase >= 1.2.1
- Apache Hadoop versions >= 2.3
- Cloudera CDH5. GeoWave tests with CDH 5.9
- Hortonworks Data Platform 2.6+
- Java Advanced Imaging and Java Image I/O are also both required to be installed on the GeoServer instance(s), as well as on the Accumulo nodes. The Accumulo support is only required for certain functions (distributed rendering), - so this may be skipped in some cases.

Example screenshots

The screenshots below are of data loaded from various attributed data sets into a GeoWave instance, processed (in some cases) by a GeoWave analytic process, and rendered by Geoserver.

GeoLife

Microsoft research has made available a trajectory data set that contains the GPS coordinates of 182 users over a three year period (April 2007 to August 2012). There are 17,621 trajectories in this data set.

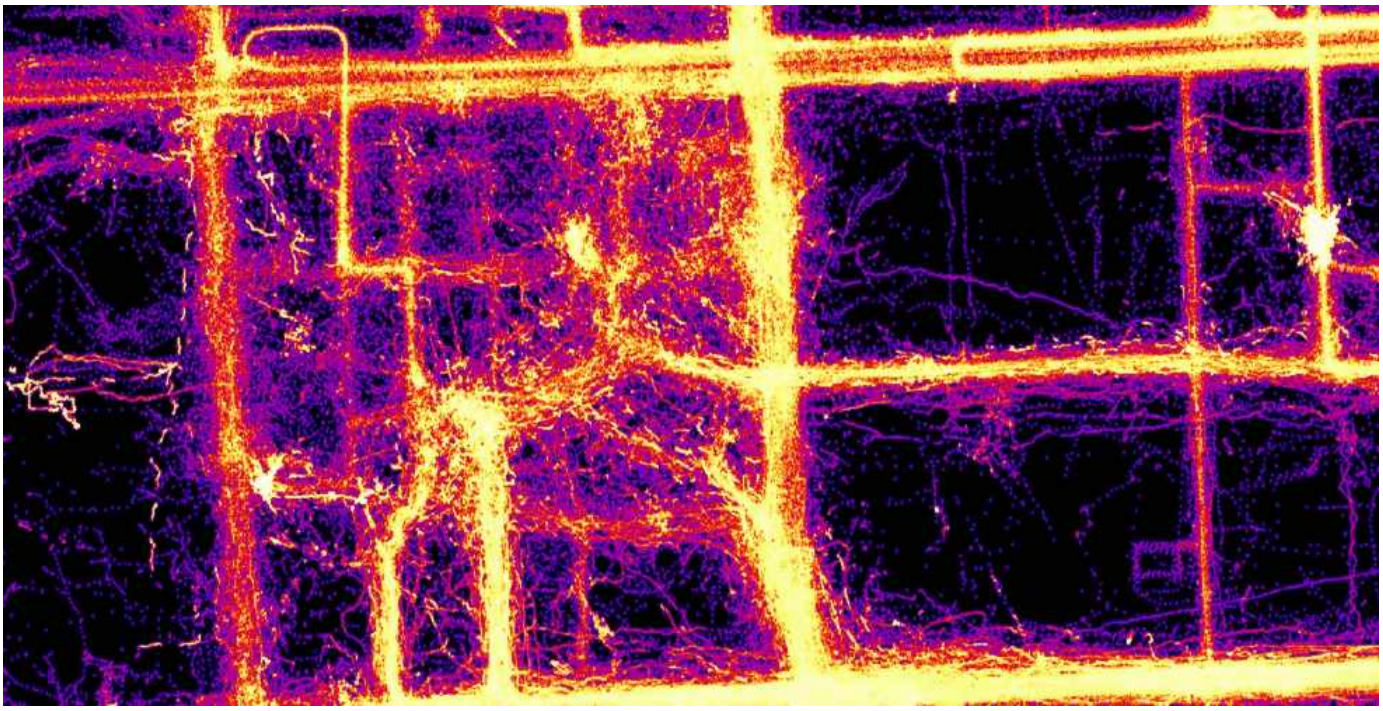
More information on this data set is available at Microsoft Research GeoLife page.

GeoLife at City Scale

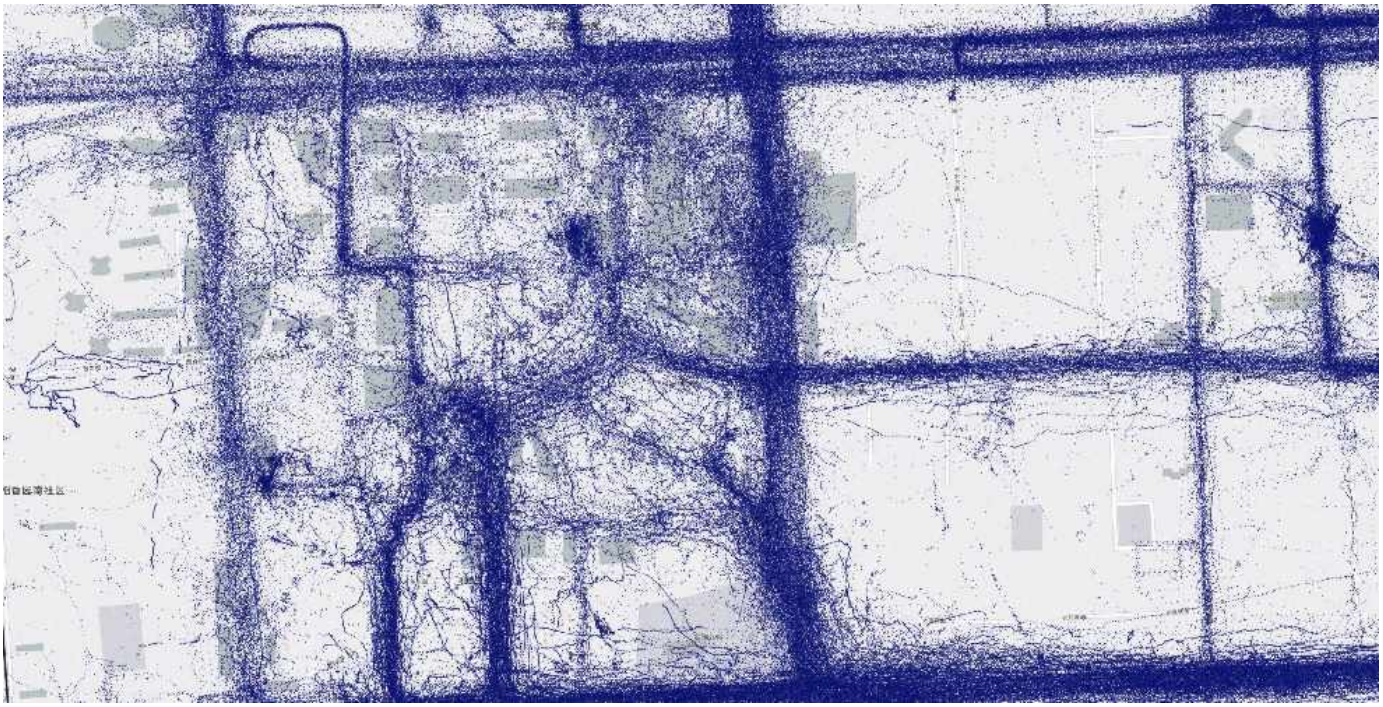
Below are renderings of GeoLife data. They display the raw points as well as the results of a GeoWave kernel density analytic. The data corresponds to Mapbox zoom level 13.

GeoLife at House Scale

This data set corresponds to a Mapbox zoom level of 15



Graphic background ©MapBox and ©OpenStreetMap



Graphic background ©MapBox and ©OpenStreetMap

OpenStreetMap GPX Tracks

The OpenStreetMap Foundation has released a large set of user contributed GPS tracks. These are about eight years of historical tracks. The data set consists of just under three billion (not trillion as some websites claim) points, or just under one million trajectories.

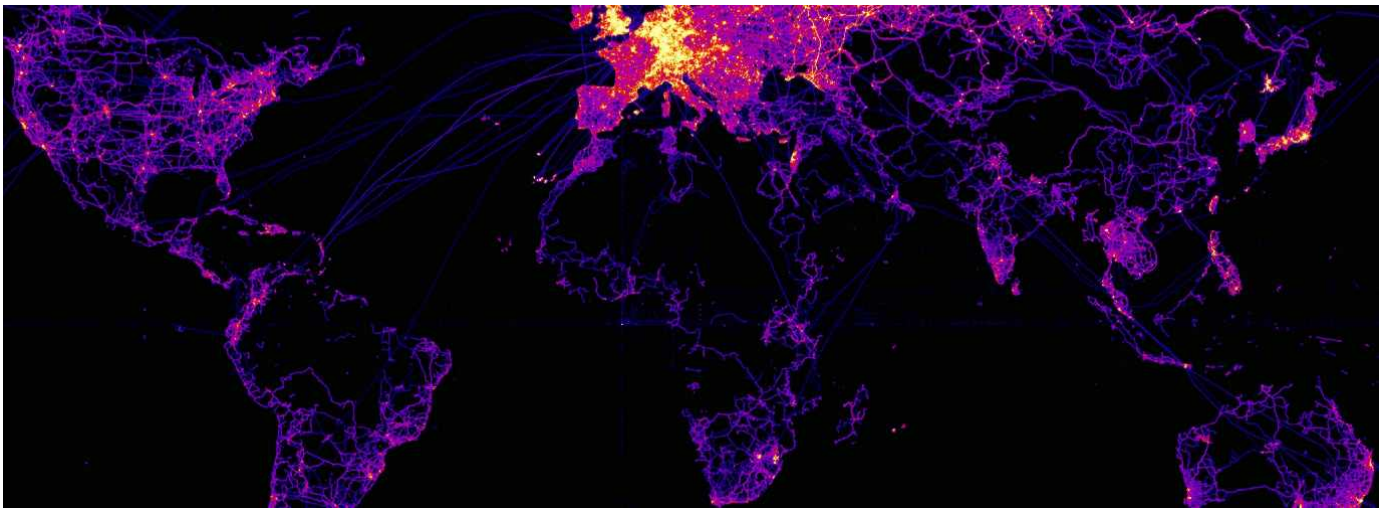
More information on this data set is available at GPX Planet page.

OSM GPX at Continent Scale

The data below corresponds to a Mapbox zoom level of 6

OSM GPX at World Scale

This data set corresponds to a Mapbox zoom level of 3



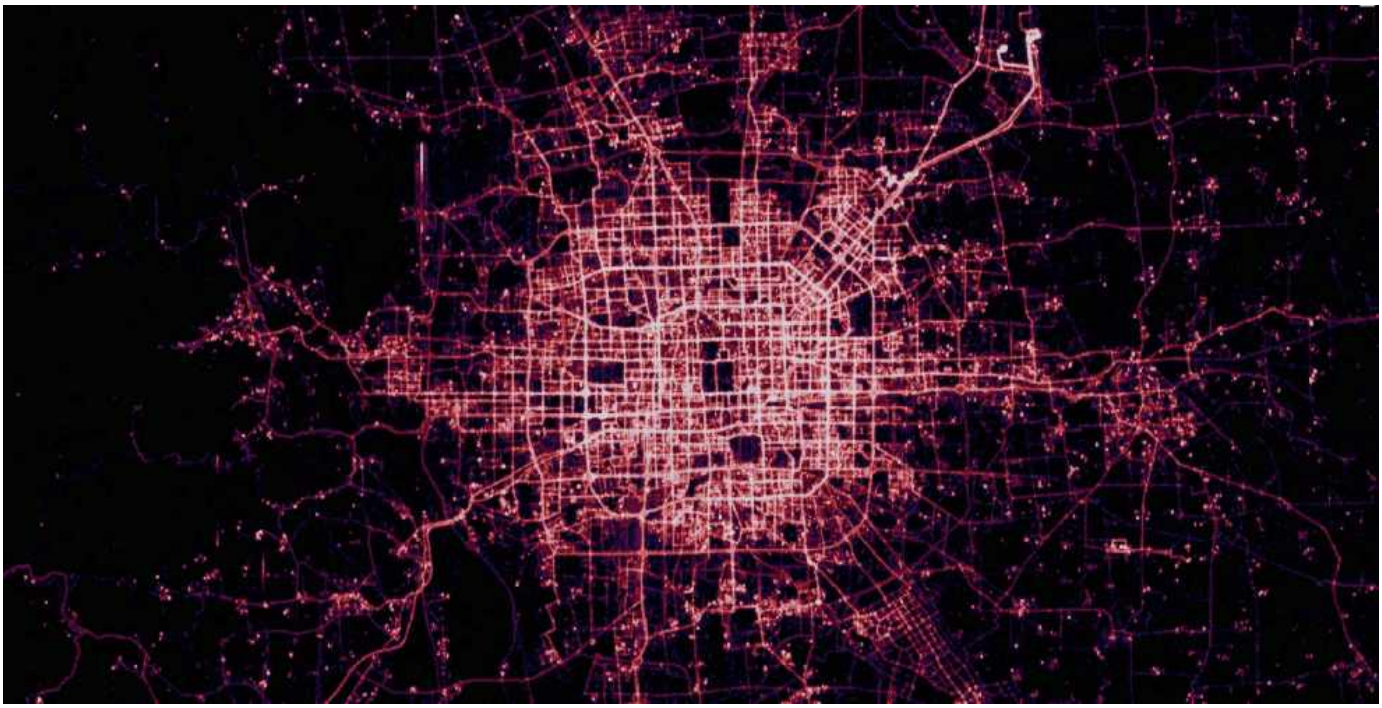
T-Drive

Microsoft research has made available a trajectory data set that contains the GPS coordinates of 10,357 taxis in Beijing, China and surrounding areas over a one week period. There are approximately 15 million points in this data set.

More information on this data set is available at: Microsoft Research T-drive page.

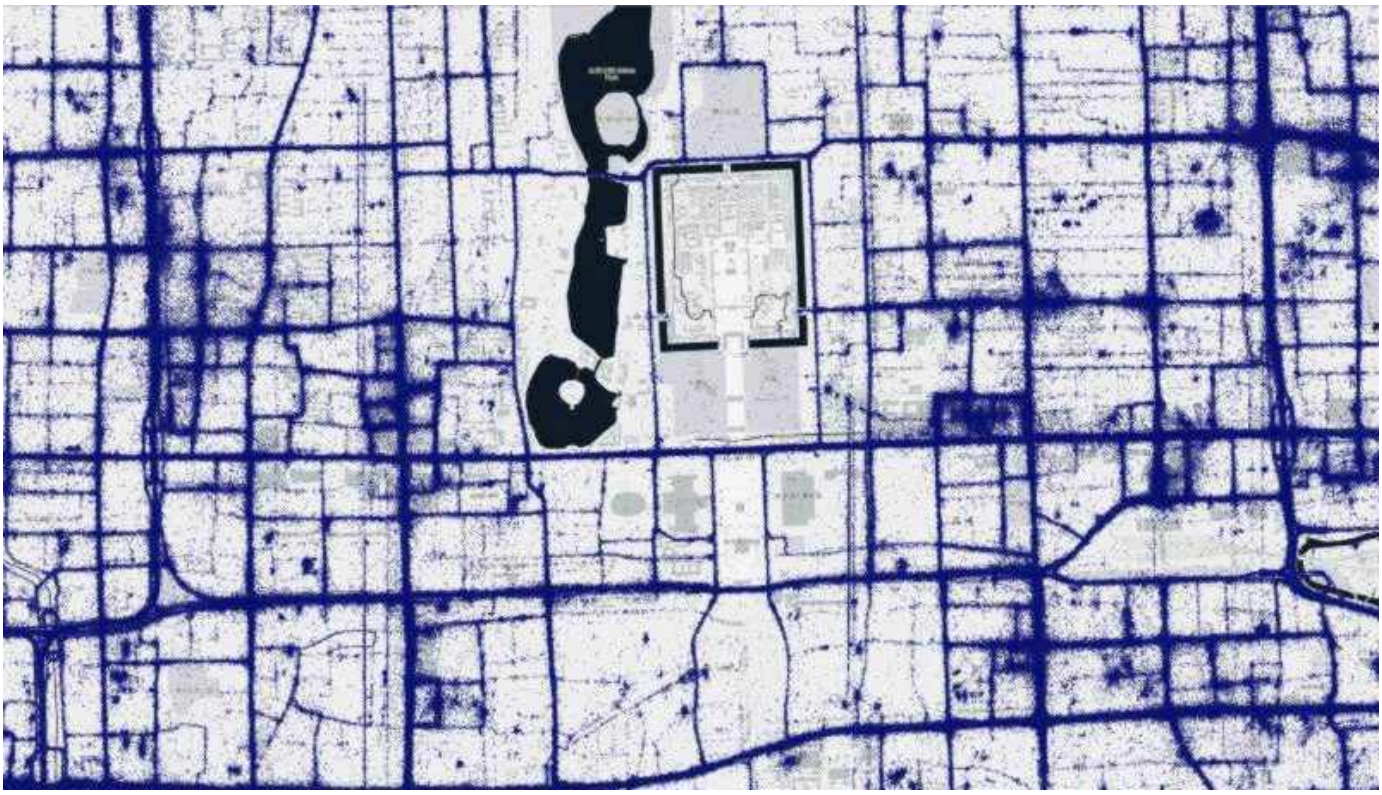
T-Drive at City Scale

Below are renderings of the t-drive data. They display the raw points along with the results of a GeoWave kernel density analytic. The data corresponds to Mapbox zoom level 12.

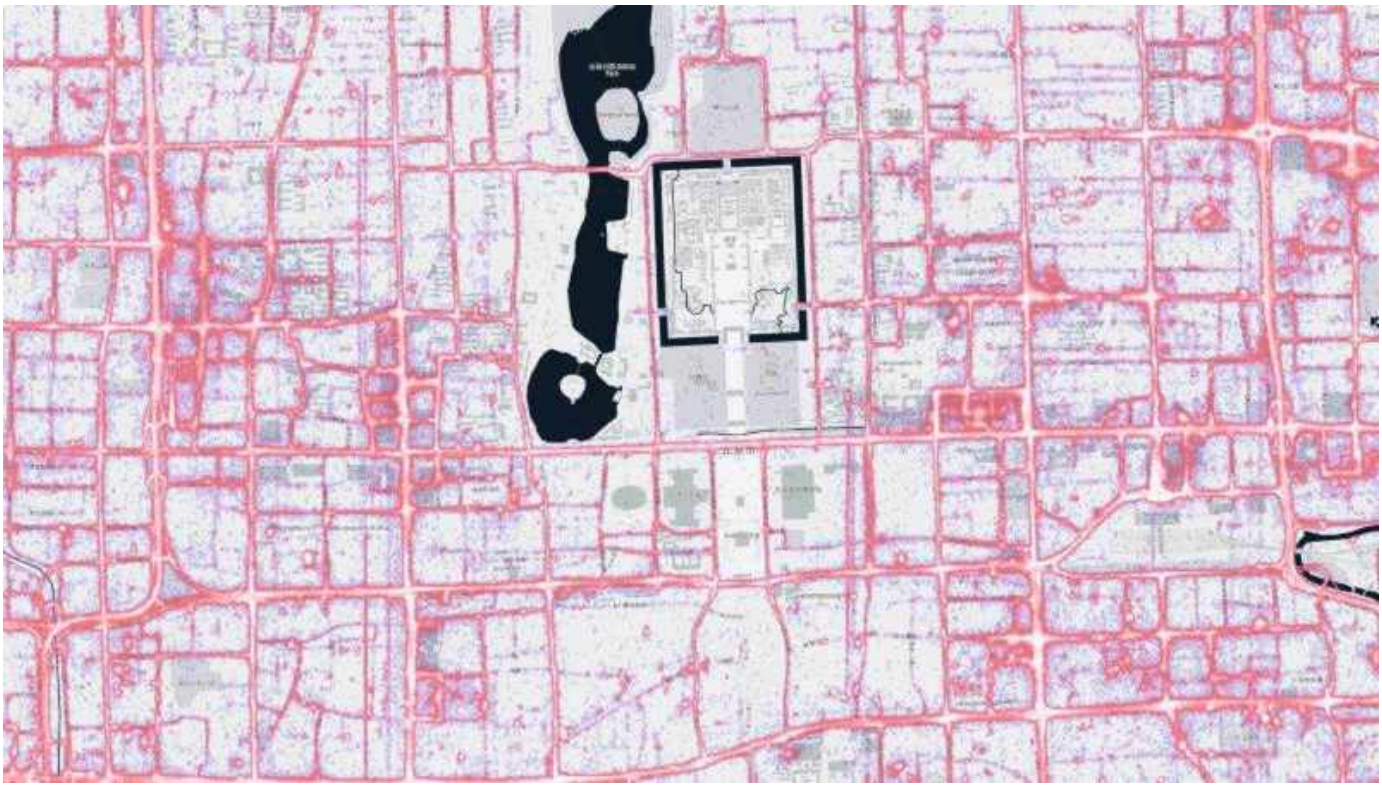


T-Drive at Block Scale

This data set corresponds to a Mapbox zoom level of 15



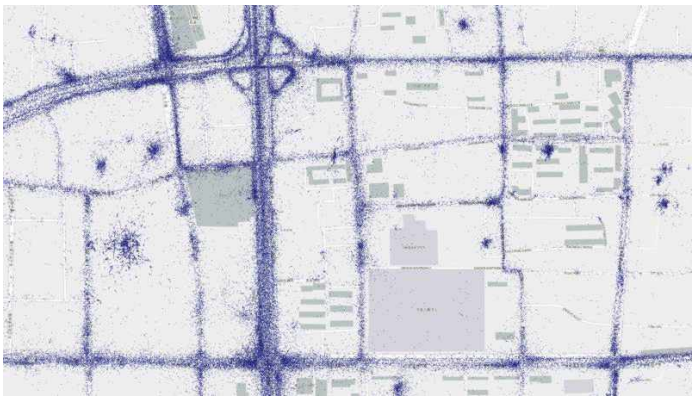
Graphic background©MapBox and ©OpenStreetMap



Graphic background©MapBox and ©OpenStreetMap

T-Drive at House Scale

This data set corresponds to a Mapbox zoom level of 17



Graphic background©MapBox and ©OpenStreetMap

Graphic background©MapBox and ©OpenStreetMap

Statistics

Adapters provide a set of statistics stored within a statistic store. The set of available statistics is specific to each adapter and the set of attributes for those data items managed by the adapter. Statistics include:

- Ranges over an attribute, including time
- Enveloping bounding box over all geometries
- Cardinality of the number of stored items
- Histograms over the range of values for an attribute
- Cardinality of discrete values of an attribute

Statistics are updated during data ingest and deletion. Range and bounding box statistics reflect the largest range over time. Those statistics are not updated during deletion. Statistics based on cardinality are updated upon deletion.

Re-computation of statistics is required in these three circumstances:

1. As indexed items are removed from the adapter store, the range and envelope statistics may lose their accuracy if the removed item contains an attribute that represents the minimum or maximum value for the population.
2. New statistics added to the statistics store after data items are ingested. These new statistics do not reflect the entire population.
3. Software changes invalidate prior stored images of statistics.

Statistics retain the same visibility constraints as the associated attributes. Thus, there is a set of statistics for each unique constraint. The statistics store answers each statistics inquiry for a given adapter with only those statistics matching the authorizations of the requester. The statistics store merges authorized statistics covering the same attribute.

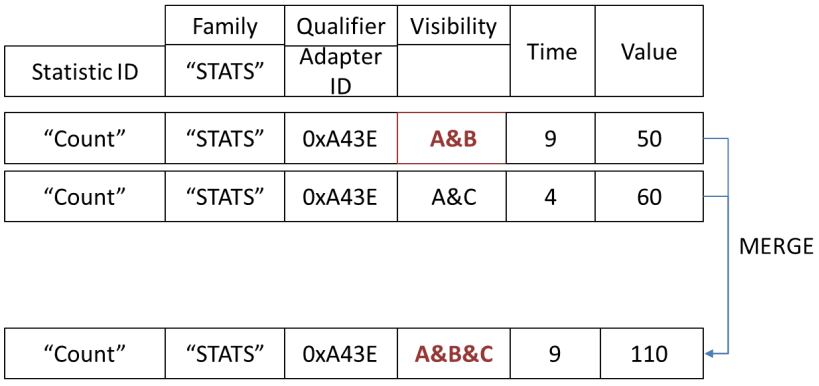
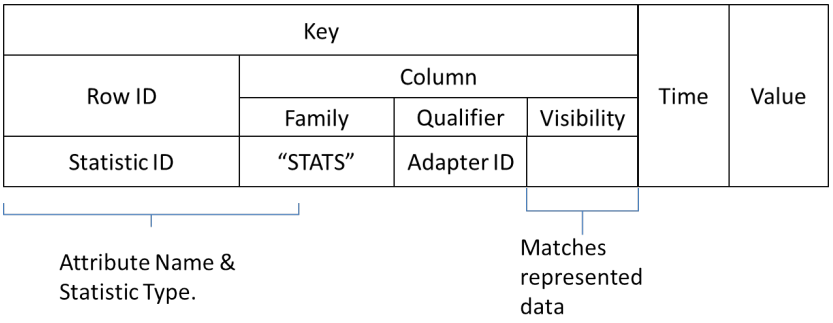


Table Structure



Tools Framework

A plugin framework (using Service Provider Interface [SPI] based injection) is provided with several input formats and utilities supported out of the box.

NOTE	This section assumes that a GeoWave build is available. Building GeoWave from source is outside the scope of this document. For details on generating a GeoWave build, please reference the GeoWave Developer Guide.

GeoWave Command Line Instructions

GeoWave comes available with several levels of command-line functionality. In this section, we will provide a high-level overview of the commands and will also outline a few of the core functions available, though for a full exhaustive list of commands, please see the GeoWave CLI Appendix.

NOTE	It is assumed that a *geowave* system command alias is registered in the terminal session being run through. To test this, type 'geowave' (no quotes) and press enter. If a list of geowave options is returned, then the system command alias is available. Otherwise, if an error similar to or containing the term, *'geowave: command not found'* is returned, the system command alias has not been registered. For details on how to register the 'geowave' system command alias, please refer to the GeoWave Developer Guide. If the alias system command is not registered, the full java command - (e.g., java -cp {GEOWAVE_HOME} {GEOWAVE_JAR}) - will need to be used in place of the 'geowave' alias.

General Usage & Options

For root-level command-line usage of GeoWave, run the 'geowave' command.

```
$ geowave
```

This will return all available commands through GeoWave along with a brief description of each.

- --config-file:

  Override configuration file (default is <home>/.geowave/config.properties).

  This flag must come after 'geowave' and before any subcommand.

- --debug:

  Verbose output. Use the debug flag to increase the debug logging output by GeoWave on the console to DEBUG. By default, it is set to WARN.

  This flag must come after 'geowave' and before any subcommand.

- --version:

  The version flag will output the build arguments that were used to build GeoWave as well as the version of the GeoWave tools jar you're using:

NOTE	Not all three options are required. All have been listed as example, though each can be used independently or as any combination of the three options.

Example:

```
$ geowave --config-file {path to configuration file override} --debug --version
```

Analytic

Commands that run mapreduce or spark processing to enhance an existing GeoWave dataset

```
$ geowave analytic
```

Store

Commands for managing GeoWave data stores

```
$ geowave store
```

Index

Commands for managing GeoWave indices

```
$ geowave index
```

Statistics

Commands for managing statistics

```
$ geowave stat
```

Config

Commands that affect local configuration only

```
$ geowave config
```

Explain

See what arguments are missing and what values will be used for GeoWave commands

```
$ geowave explain
```

GeoServer

Commands that manage geoserver data stores and layers

```
$ geowave gs
```

Accumulo

Accumulo utility commands

```
$ geowave util accumulo
```

HBase

HBase utility commands

```
$ geowave util hbase
```

Python

Utility commands for python integration

```
$ geowave util python
```

Help

The help command will show arguments and their defaults. It can be prepended to any GeoWave command. If you use it while also specifying a sub-command and its arguments, that command's help information will be displayed:

```
$ geowave help <command> <subcommand>
```

Ingest

Commands that ingest data directly into GeoWave or stage data to be ingested into GeoWave

```
$ geowave ingest
```

Landsat

Operations to analyze, download, and ingest Landsat 8 imagery publicly available on AWS

```
$ geowave util landsat
```

OSM

Operations to ingest OSM nodes, ways and relations to GeoWave

```
$ geowave util osm
```

Raster

Operations to perform transformations on raster data in GeoWave

```
$ geowave raster
```

Vector

Vector data operations

```
$ geowave vector
```

Ingest

Overview

In addition to the raw data to ingest, the ingest process requires an adapter to translate the native data into a format that can be persisted into the data store. Also, the ingest process requires an Index that is a definition of all the configured parameters that define how data is translated to Row IDs (how it is indexed). It also includes what common fields need to be maintained within the table to be used by fine-grained and secondary filters.

There are various ways to ingest data into a GeoWave store. The standard localToGW command is used to ingest files from a local file system or from an AWS S3 bucket into GeoWave in a single process. For a distributed ingest (recommended for larger datasets) the sparkToGW and mrToGW commands can be used. Ingests can also be performed directly from HDFS or utilizing Kafka.

The full list of GeoWave ingest commands can be found in the GeoWave CLI Appendix.

For examples and other details of running ingest commands, please be sure to check out the GeoWave QuickStart Guide Examples.

Ingest Plugins

The *geowave* command line utility comes with several plugins out of the box. You can list the available plugins that are registered with your commandline tool.

```
$ geowave ingest listplugins
```

NOTE You can add more by simply copying a desired plugin into the */usr/local/geowave/tools/plugins* directory.

The above listplugins command should yield a result similar to what is shown below.

```
Available index types currently registered as plugins:

  spatial_temporal:
    This dimensionality type matches all indices that only require Geometry and Time.

  spatial:
    This dimensionality type matches all indices that only require Geometry.

Available ingest formats currently registered as plugins:

  twitter:
    Flattened compressed files from Twitter API

  geotools-vector:
    all file-based vector datastores supported within geotools

  geolife:
    files from Microsoft Research GeoLife trajectory data set

  gdelt:
    files from Google Ideas GDELT data set

  stanag4676:
    xml files representing track data that adheres to the schema defined by STANAG-4676

  geotools-raster:
    all file-based raster formats supported within geotools

  gpx:
    xml files adhering to the schema of gps exchange format

  tdrive:
    files from Microsoft Research T-Drive trajectory data set

  avro:
    This can read an Avro file encoded with the SimpleFeatureCollection schema.  This
schema is also used by the export tool, so this format handles re-ingesting exported
datasets.

Available datastores currently registered:

  accumulo:
    A GeoWave store backed by tables in Apache Accumulo

  bigtable:
    A GeoWave store backed by tables in Google's Cloud BigTable
```

```
  hbase:
    A GeoWave store backed by tables in Apache HBase
```

Ingest Statistics and Time Dimension Configuration

The available plugins for vector support adjustments to their configuration via the command line. The system property 'SIMPLE_FEATURE_CONFIG_FILE' may be assigned to the name of a locally accessible JSON file defining the configuration.

Example

```
$ GEOWAVE_TOOL_JAVA_OPT="-DSIMPLE_FEATURE_CONFIG_FILE=myconfigfile.json"
$ geowave ingest localtogw ./ingest mystore myindex
```

Configuration consists of several parts:

1. Selecting temporal attributes for a temporal index
2. Assigning to each attribute the type of statistics to be captured within the Statistics Store
3. Determining which attributes should be indexed in a secondary index
4. Determining which attribute contains visibility information for other attributes
5. Setting the names of the indices to update in WFS-T transactions via the GeoServer plug-in

The JSON file is made up of configurations. Each configuration is defined by a class name and a set of attributes. Configurations are grouped by the Simple Feature Type name.

Temporal Configuration

There are three attributes for the temporal configuration:

1. timeName
2. startRangeName
3. endRangeName

These attributes are associated with the name of a simple feature type attribute that references a time value. To index by a single time attribute, set `timeName` to the name of the single attribute. To index by a range, set both `startRangeName` and `endRangeName` to the names of the simple feature type attributes that define start and end time values.

Statistics Configuration

Each simple feature type attribute may have several assigned statistics. Bounding box and range statistics are automatically captured for geometry and temporal attributes.

Attribute Type	Statistic Name	Statistic Configuration Attributes (with default values)	Statistic Class
Numeric	Fixed Bin Histogram	minValue=-∞, maxValue=∞, bins=32	org.locationtech.geowave.adapter.vector.stats.FeatureFixedBinNumericStatistics$FeatureFixedBinConfig
	Dynamic Histogram		org.locationtech.geowave.adapter.vector.stats.FeatureNumericHistogramStatistics$FeatureNumericHistogramConfig
	Numeric Range		org.locationtech.geowave.adapter.vector.stats.FeatureNumericRangeStatistics$FeatureNumericRangeConfig
String	Count Min Sketch	errorFactor=0.001, probabilityOfCorrectness=0.98	org.locationtech.geowave.adapter.vector.stats.FeatureCountMinSketchStatistics$FeatureCountMinSketchConfig
	Hyper Log Log	precision=16	org.locationtech.geowave.adapter.vector.stats.FeatureHyperLogLogStatistics$FeatureHyperLogLogConfig

Visibility Configuration

Visibility configuration has two attributes: the visibility manager class and the visibility attribute name.

A Visibility Manager extends *org.locationtech.geowave.core.store.data.visibility.VisibilityManagement*. An instance of this class interprets the contents of a visibility attribute, within a simple feature, to determine the visibility constraints of the other attributes in that simple feature. The default visibility management class is *org.locationtech.geowave.adapter.vector.plugin.visibility.JsonDefinitionColumnVisibilityManagement*.

Secondary Index Configuration

Secondary Index Configurations is made up of one of three classes:

- *org.locationtech.geowave.adapter.vector.index.NumericSecondaryIndexConfiguration*
- *org.locationtech.geowave.adapter.vector.index.TemporalSecondaryIndexConfiguration*
- *org.locationtech.geowave.adapter.vector.index.TextSecondaryIndexConfiguration*

Each of this configurations maintains a set of simple feature attribute names to index in a secondary index.

Primary Index Identifiers

The class *org.locationtech.geowave.adapter.vector.index.SimpleFeaturePrimaryIndexConfiguration* is used to maintain the configuration of primary indices used for adding or updating simple features via the GeoServer plug-in (FeatureWriter).

Example

```
{
  "configurations": {
    "myFeatureTypeName" : [
      {
        "@class" :
"org.locationtech.geowave.adapter.vector.utils.TimeDescriptors$TimeDescriptorConfiguratio
n",
        "startRangeName":null,
        "endRangeName":null,
        "timeName":"captureTime"
       },
       {
         "@class":
"org.locationtech.geowave.adapter.vector.index.NumericSecondaryIndexConfiguration",
         "attributes" : ["pop"]
       },
       {
         "@class":
"org.locationtech.geowave.adapter.vector.plugin.visibility.VisibilityConfiguration",
         "attributeName" : "vis"
       },
       {
        "@class":
"org.locationtech.geowave.adapter.vector.index.SimpleFeaturePrimaryIndexConfiguration",
         "indexNames": ["SPATIAL_IDX"]
       }
       {
         "@class" :
"org.locationtech.geowave.adapter.vector.stats.StatsConfigurationCollection$SimpleFeature
StatsConfigurationCollection",
         "attConfig" : {
           "population" : {
             "configurationsForAttribute" :  [
             {
               "@class" :
"org.locationtech.geowave.adapter.vector.stats.FeatureFixedBinNumericStatistics$FeatureFi
xedBinConfig",
               "bins" : 24
             }
             ]
           },
          "country" : {
            "configurationsForAttribute" :  [
             {
               "@class" :
"org.locationtech.geowave.adapter.vector.stats.FeatureCountMinSketchStatistics$FeatureCou
ntMinSketchConfig",
```

```
              "probabilityOfCorrectness" : 0.98,
              "errorFactor" :0.001
            },
            {
              "@class" :
"org.locationtech.geowave.adapter.vector.stats.FeatureHyperLogLogStatistics$FeatureHyperL
ogLogConfig"
            }
           ]
         }
       }
     }
   ]
  }
}
```

Analytics

Overview

Analytics embody algorithms tailored to geospatial data. Most analytics leverage Hadoop MapReduce for bulk computation. Results of analytic jobs consist of vector or raster data stored in GeoWave. The analytics infrastructure provides tools to build algorithms in Spark. For example, a Kryo serializer/deserializer enables exchange of SimpleFeatures and the GeoWaveInputFormat supplies data to the Hadoop RDD.

NOTE

GeoWaveInputFormat does not remove duplicate features that reference lines and/or polygons spanning multiple index regions. If working with duplication in time ranges, please take a look at the GeoWave GeoWaveDedupeJobRunner.

It is also important to note that, while duplication can arise as an issue, though if using the XZ-Order SFC (which is used by default for lines and polygons), it does not require duplication. The only time there should be a duplication issue is if you are indexing a time range and the time range spans multiple periods based on your temporal binning strategy. The default periodicity is "YEAR," meaning that a row must be duplicate under default spatial-temporal indexing if you are trying to index a time range that crosses December 31 23:59:59.999 and January 1 00:00:00 of the following year.

The following algorithms are provided.

Name	Description
KMeans++	A K-Means implementation to find K centroids over the population of data. A set of preliminary sampling iterations find an optimal value of K and the initial set of K centroids. The algorithm produces K centroids and their associated polygons. Each polygon represents the concave hull containing all features associated with a centroid. The algorithm supports drilling down multiple levels. At each level, the set centroids are determined from the set of features associated the same centroid from the previous level.
KMeans Jump	Uses KMeans++ over a range of k, choosing an optimal k using an information theoretic based measurement. Example Usage: *yarn jar geowave-tools.jar analytic kmeansjump*
KMeans Parallel	Performs a KMeans Parallel Cluster Example Usage: *yarn jar geowave-tools.jar analytic kmeansparallel*
DBScan	The Density Based Scanner algorithm produces a set of convex polygons for each region meeting density criteria. Density of region is measured by a minimum cardinality of enclosed features within a specified distance from each other. Example Usage: *yarn jar geowave-tools.jar analytic dbscan*

Name	Description
Nearest Neighbors	An infrastructure component that produces all the neighbors of a feature within a specific distance. Example Usage: *yarn jar geowave-tools.jar analytic nn*

NOTE	Building/Developing GeoWave analytics is outside the scope of this document. For details around how to build the GeoWave source and tools, please refer to the GeoWave Developer Guide.

Running

The GeoWave analytical tools are made available through MapReduce and Spark implementations. To run the tools, use the yarn or hadoop system APIs, as outlined below.

```
yarn jar geowave-tools.jar analytic <algorithm> <options> <store>
```

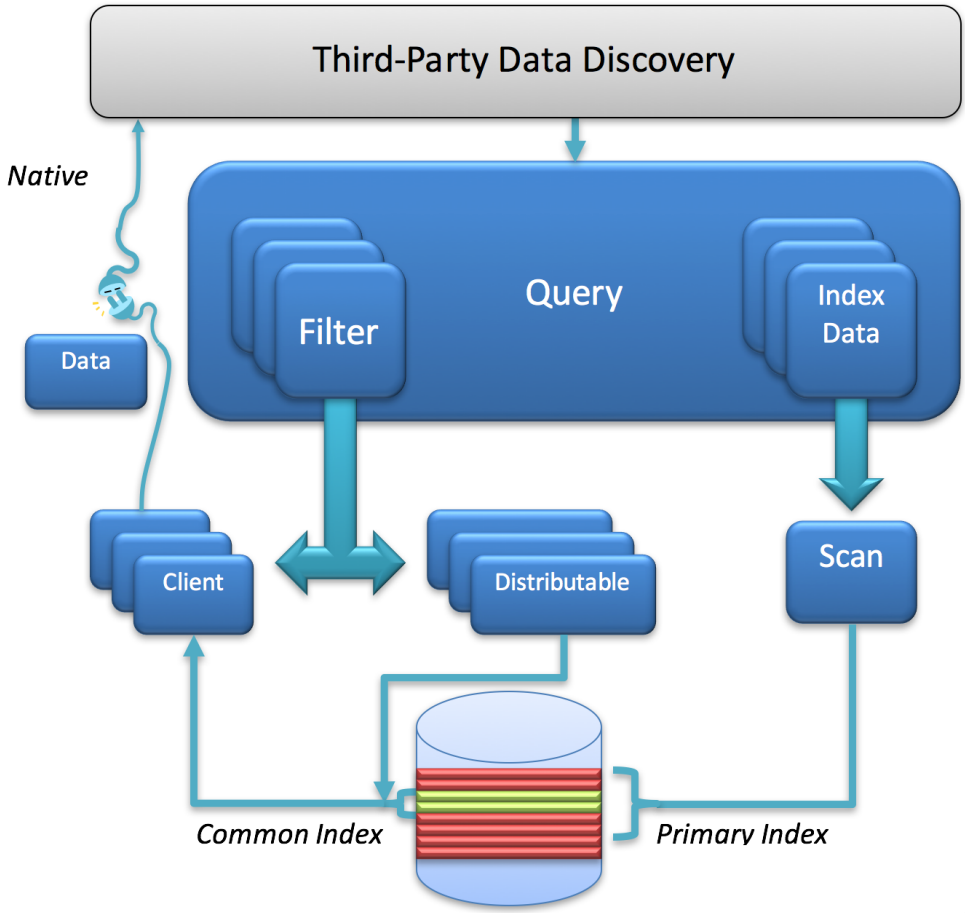
The above command will execute <algorithm> (such as dbscan), sourcing the data from the <store> datastore (see store add).

Analytic Commands

The full list of GeoWave analytic commands, and any details associated with each, can be found in the GeoWave CLI Appendix.

Query

Overview



A query in GeoWave currently consists of a set of ranges on the dimensions of the primary index. Up to three dimensions, plus temporal optionally, can take advantage of any complex OGC geometry for the query window. For dimensions of four or greater the query can only be a set of ranges on each dimension, e.g., hyper-rectangle, etc.

The query geometry is decomposed by GeoWave into a series of ranges on a one dimensional number line - based on a compact Hilbert space filling curve (SFC) ordering. These ranges are sent through an Accumulo batch scanner to all the tablet servers. These ranges represent the coarse grain filtering.

At the same time, the query geometry has been serialized and sent to custom Accumulo iterators. These iterators then do a second stage filtering on each feature for an exact intersection test. Only if the stored geometry and the query geometry intersect does the processing chain continue.

A second order filter can then be applied - this is used to remove features based on additional attributes, typically time or other feature attributes. These operators can only exclude items from the set defined by the range - they cannot include additional features. Think "AND" operators not "OR".

A final filter is possible on the client set after all the returned results have been aggregated together. Currently, this is only used for final de-duplication. Whenever possible, the distributed filter options should be used as it splits the work load among all the tablet servers.

Third Party

GeoServer

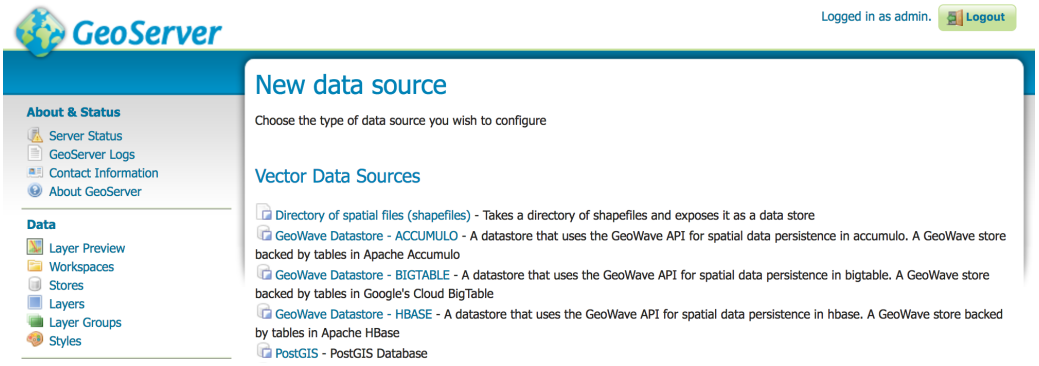
This section will outline the various aspects of the GeoServer tools that are relevant to GeoWave capabilities. While GeoServer is a third-party tool that integrates with GeoWave, it is important to note that this is not meant to be an exhaustive guide to GeoServer, though it's more of an overview and integration guide. For official GeoServer documentation and how-to guides, please reference the GeoServer documentation guides.

NOTE	This section assumes that a GeoWave GeoServer plugin has already been created. If this is not the case and a plugin needs to be created, please refer to the GeoServer plugin section.

GeoWave supports both raster images and vector data exposed through Geoserver.

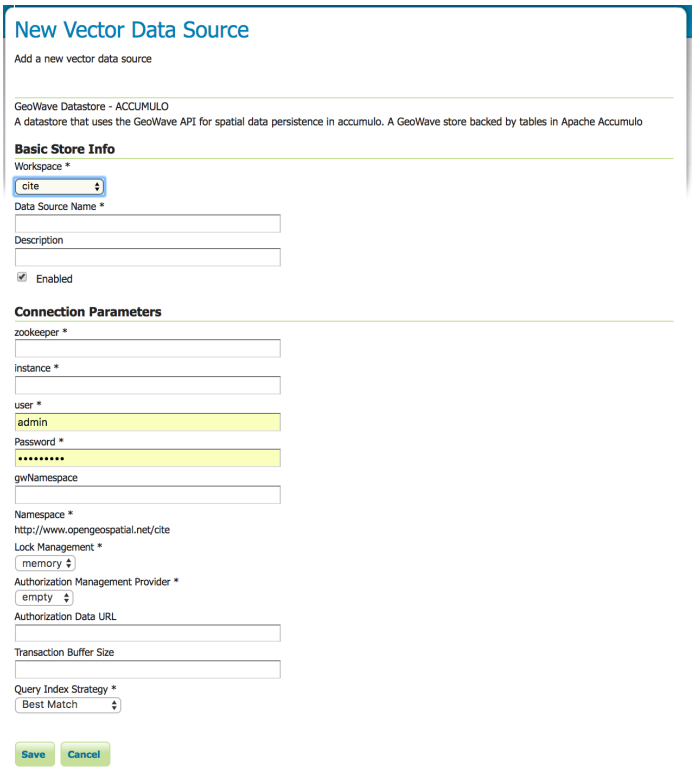
WFS-T

GeoWave supports WFS-T for vector data by extending GeoTools. After following the deployment steps, GeoWave appears as the data store types called 'GeoWave Datastore - ACCUMULO' and 'GeoWave Datastore - HBASE'.



Accumulo Datastore Plugin

On the Geowave Datastore - ACCUMULO creation tab, the system prompts for the following connection parameters.

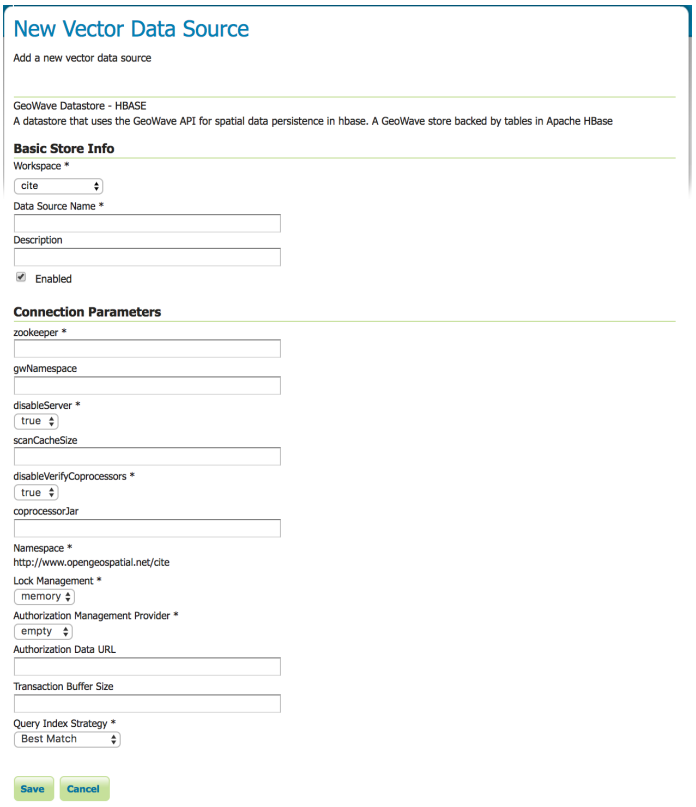


Name	Description	Constraints
zookeeper	Comma-separated list of Zookeeper host and port	Host and port are separated by a colon (host:port)
instance	The Accumulo tablet server's instance name	The name matches the one configured in Zookeeper
user	The Accumulo user name	The user should have administrative privileges to add and remove authorized visibility constraints
password	Accumulo user's password	
gwNamespace	The table namespace associated with this Accumlo data store	
Lock Management	Select one from a list of lock managers	Zookeeper is required with a multiple Geoserver architecture
Authorization Management Provider	Select from a list of providers	
Authorization Data URL	The URL for an external supporting service or configuration file	The interpretation of the URL depends on the selected provider

Name	Description	Constraints
Query Index Strategy	The pluggable query strategy to use for querying geowave tables - a reasonable default will be used if not supplied	

HBase Datastore Plugin

On the Geowave Datastore - HBASE creation tab, the system prompts for the following connection parameters.



Name	Description	Constraints
enableCustomFilters	Allows for the use of custom filters	Defaults to true
zookeeper	Comma-separated list of Zookeeper host and port	Host and port are separated by a colon (host:port)
enableCoprocessors	Allows for the use of HBase co-processors	Defaults to true
gwNamespace	The table namespace associated with this Hbase data store	
verifyCoprocessors		Defaults to true

Name	Description	Constraints
Lock Management	Select one from a list of lock managers	Zookeeper is required with a multiple Geoserver architecture
Authorization Management Provider	Select from a list of providers	
Authorization Data URL	The URL for an external supporting service or configuration file	The interpretation of the URL depends on the selected provider
Query Index Strategy	The pluggable query strategy to use for querying geowave tables - a reasonable default will be used if not supplied.	

GeoServer Configuration

GeoWave can be configured for a GeoServer connection through the GeoServer config command line interface (CLI) operation.

```
geowave config geoserver {GEOSERVER URL} --user {USERNAME} --pass {PASSWORD}
```

Argument	Required	Description
--url	True	GeoServer URL (for example http://localhost:8080/geoserver), or simply host:port and appropriate assumptions are made
--username	True	GeoServer User
--password	True	GeoServer Password - Refer to the password security section for more details and options
--workspace	False	GeoServer Default Workspace

GeoWave supports connecting to GeoServer through both HTTP and HTTPS (HTTP + SSL) connections. If connecting to GeoServer through an HTTP connection (e.g., http://localhost:8080/geoserver), the command above is sufficient.

GeoServer SSL Connection Properties

If connecting to GeoServer through a Secure Sockets Layer (SSL) connection over HTTPS (e.g., https://localhost:8443/geoserver), some additional configuration options need to be specified, in order for the system to properly establish the secure connection's SSL parameters. Depending on the particular SSL configuration through which the GeoServer server is being connected, you will need to specify which parameters are necessary.

NOTE	Not all SSL configuration settings may be necessary, as it depends on the setup of the SSL connection through which GeoServer is hosted. Contact your GeoServer administrator for SSL connection related details.

SSL Argument	Description
--sslKeyManagerAlgorithm	Specify the algorithm to use for the keystore.
--sslKeyManagerProvider	Specify the key manager factory provider.
--sslKeyPassword	Specify the password to be used to access the server certificate from the specified keystore file. - Refer to the password security section for more details and options.
--sslKeyStorePassword	Specify the password to use to access the keystore file. - Refer to the password security section for more details and options.

SSL Argument	Description
--sslKeyStorePath	Specify the absolute path to where the keystore file is located on system. The keystore contains the server certificate to be loaded.
--sslKeyStoreProvider	Specify the name of the keystore provider to be used for the server certificate.
--sslKeyStoreType	The type of keystore file to be used for the server certificate, e.g., JKS (Java KeyStore).
--sslSecurityProtocol	Specify the Transport Layer Security (TLS) protocol to use when connecting to the server. By default, the system will use TLS.
--sslTrustManagerAlgorithm	Specify the algorithm to use for the truststore.
--sslTrustManagerProvider	Specify the trust manager factory provider.
--sslTrustStorePassword	Specify the password to use to access the truststore file. - Refer to the password security section for more details and options
--sslTrustStorePath	Specify the absolute path to where truststore file is located on system. The truststore file is used to validate client certificates.
--sslTrustStoreProvider	Specify the name of the truststore provider to be used for the server certificate.
--sslTrustStoreType	Specify the type of key store used for the truststore, e.g., JKS (Java KeyStore).

WFS-T

Transactions are initiated through a Transaction operation, that contains inserts, updates, and deletes to features. WFS-T supports feature locks across multiple requests by using a lock request followed by subsequent use of a provided *Lock ID*. The GeoWave implementation supports transaction isolation. Consistency during a commit is not fully supported. Thus, a failure during a commit of a transaction may leave the affected data in an intermediary state. Some deletions, updates, or insertions may not be processed in such a case. The client application must implement its own compensation logic upon receiving a commit-time error response. As expected with Accumulo, operations on a single feature instances are atomic.

Inserted features are buffered prior to commit. The features are bulk fed to the data store when the buffer size is exceeded and when the transaction is committed. In support of atomicity and isolation, flushed features, prior to commit, are marked in a transient state and are only visible to the controlling transaction. Upon commit, these features are 'unmarked'. The overhead incurred by this operation is avoided by increasing the buffer size to avoid pre-commit flushes.

Lock Management

Lock management supports life-limited locks on feature instances. The only supported lock manager is in-memory, which is suitable for single Geoserver instance installations.

Index Selection

Data written through WFS-T is indexed within a single index. The adapter inspects existing indices, finding one that matches the data requirements. A geo-temporal index is chosen for features with temporal attributes. The adapter creates a geospatial index upon failure of finding a suitable index. A geotemporal index is not created, regardless of the existence of temporal attributes. Currently, geotemporal indices lead to poor performance for queries requesting vectors over large spans of time.

Authorization Management

Authorization Management provides the set of credentials compared against the security labels attached to each cell. Authorization Management determines the set of authorizations associated with each WFS-T request. The available Authorization Management strategies are registered through the Server Provider model, within the file

`META-INF/services/org.locationtech.geowave.vector.auth.AuthorizationFactorySPI`.

The provided implementations include the following

- Empty - Each request is processed without additional authorization.
- JSON - The requester user name, extracted from the Security Context, is used as a key to find the user's set of authorizations from a JSON file. The location of the JSON file is determined by the associated *Authorization Data URL* (e.g., */opt/config/auth.json*). An example of the contents of the JSON file is given below.

```
{
  "authorizationSet": {
     "fred" : ["1","2","3"],
     "barney" : ["a"]
  }
}
```

Fred has three authorization labels. Barney has one.

Visibility Management

Visibility constraints, applied to feature instances during insertions, are ultimately determined in `org.locationtech.geowave.store.data.field.FieldWriter`. There are writers for each supported data type in Geoserver. By default, the set visibility expression attached to each feature property is empty. Visibility Management supports selection of a strategy by wrapping each writer to provide visibility. This alleviates the need to extend the type specific FieldWriters.

The visibility management strategy is registered through the Java Service Provider Interface (SPI) model, within the file `META-INF/services/org.locationtech.geowave.vector.plugin.visibility.ColumnVisibilityManagement`. The only provided implementation is the `JsonDefinitionColumnVisibilityManagement`. The implementation expects a property within each feature instance to contain a JSON string describing how to set the visibility for each property of the feature instance. This approach allows each instance to determine its own visibility criteria.

Each name/value pair within the JSON structure defines the visibility for the associated feature property with the same name. In the following example, the *geometry* property is given a visibility `S` and the eventName is given a visibility `TS`.

```
{ "geometry" : "S", "eventName": "TS" }
```

JSON attributes can be regular expressions matching more than one feature property name. In the example, all properties except for those that start with 'geo' have visibility `TS`.

```
{ "geo.*" : "S", ".*" : "TS" }
```

The order of the name/value pairs must be considered if one rule is more general than another, as shown in the example. The rule `. matches all properties. The more specific rule geo.` must be ordered first.

The system extracts the JSON visibility string from a feature instance property named `GEOWAVE_VISIBILITY`. Selection of an alternate property is achieved by setting the associated attribute descriptor 'visibility' to the boolean value TRUE.

Installation

Standalone Installers

GeoWave provides installers to access the commandline tools. It uses a multi-platform installer builder, Install4J. The installers can be downloaded here:

- Windows
- Mac
- Linux

Installation from RPM

Overview

There is a public GeoWave RPM Repo available with the distribution packages and vendors listed below. As you'll need to coordinate a restart of Accumulo to pick up changes to the GeoWave iterator classes the repos default to be disabled so you can keep auto updates enabled. When ready to do an update, simply add `--enablerepo=geowave` to your command. The packages are built for a number of different hadoop distributions (Cloudera, Hortonworks, and Apache). The RPMs have the vendor name embedded as the second portion of the rpm name (geowave-apache-accumulo, geowave-hdp2-accumulo, geowave-cdh5-accumulo, geowave-apache-hbase, etc.)

Examples

Use the GeoWave repo RPM to configure a host and search for GeoWave RPMs to install.

NOTE	Several of the RPMs (accumulo, hbase, jetty, single-host, and tools) are both GeoWave version and vendor version specific. In the examples, below the rpm name geowave-$VERSION-VENDOR_VERSION should be adjusted as needed.

Currently supported distribution vendors through GeoWave include:

Distribution Vendor	Vendor Abbreviation
Apache	apache
Cloudera	cdh5
Hortonworks	hdp2

```
rpm -Uvh http://s3.amazonaws.com/geowave-rpms/release/noarch/geowave-repo-1.0-
3.noarch.rpm

# To search for GeoWave packages for a specific distribution
yum --enablerepo=geowave search geowave-$VERSION-$VENDOR-*

# To install a specific GeoWave package on a host (probably a namenode)
yum --enablerepo=geowave install geowave-$VERSION-$VENDOR-$PACKAGE

# Update all packages for a specific venfors distribution
yum --enablerepo=geowave install geowave-$VERSION-$VENDOR-*
```

GeoWave RPMs

Name	Description
geowave-$VERSION-$VENDOR-accumulo	Accumulo Components
geowave-$VERSION-$VENDOR-hbase	HBase Components
geowave-$VERSION-core	Core (home directory and geowave user)
geowave-$VERSION-docs	Documentation (HTML, PDF and man pages)
geowave-$VERSION-$VENDOR-tools	Command Line Tools (ingest, etc.)
geowave-$VERSION-$VENDOR-gwtomcat	GeoServer components installed into /usr/local/geowave/geoserver and available at, e.g., http://localhost:8080/geoserver/web
geowave-$VERSION-puppet	Puppet Scripts
geowave-$VERSION-$VENDOR-single-host	All GeoWave components installed on a single host (sometimes useful for development)
geowave-repo	GeoWave RPM Repo config file
geowave-repo-dev	GeoWave Development RPM Repo config file

NOTE

- $VERSION: Version of GeoWave source, e.g., 1.0.0
- $VENDOR: Distribution vendor abbreviation - from vendors table above, e.g., apache, cdh5, hdp2
- $PACKAGE: Package - see RPMs table above, i.e. accumulo, hbase, jetty, single-host, tools

For all vendors or packages for a particular version, replace with * (e.g., 'geowave-1.0.0-apache-*') for all apache apache packages in 1.0.0 version of GeoWave.

Note that only accumulo, hbase, jetty, tools, and single-host are vendor package rpm's. All others are version RPMs.

RPM Installation Notes

RPM names contain the version in the name so support concurrent installations of multiple GeoWave and/or vendor versions. A versioned /usr/local/geowave-$GEOWAVE_VERSION-$VENDOR_VERSION directory is linked to /usr/local/geowave using alternatives ex: /usr/local/geowave → /usr/local/geowave-0.9.3-hdp2, but there could also be another /usr/local/geowave-0.9.2.1-cdh5 still installed but not the current default.

View geowave-home installed and default using alternatives

```
alternatives --display geowave-home
geowave-home - status is auto.
 link currently points to /usr/local/geowave-0.9.3-hdp2
/usr/local/geowave-0.9.3-hdp2 - priority 90
/usr/local/geowave-0.9.2.1-cdh5 - priority 89
Current `best' version is /usr/local/geowave-0.9.3-hdp2.
```

geowave--accumulo:*

This RPM will install the GeoWave Accumulo iterator into the local file system and then upload it into HDFS using the `hadoop fs -put` command. This means of deployment requires that the RPM is installed on a node that has the correct binaries and configuration in place to push files to HDFS, like your namenode. We also need to set the ownership and permissions correctly within HDFS and as such, need to execute the script as a user that has superuser permissions in HDFS. This user varies by Hadoop distribution vendor. If the Accumulo RPM installation fails, check the install log located at `/usr/local/geowave/accumulo/geowave-to-hdfs.log` for errors. The script can be re-run manually if there was a problem that can be corrected like the HDFS service was not started. If a non-default user was used to install Hadoop you can specify a user that has permissions to upload with the --user argument `/usr/local/geowave/accumulo/deploy-to-geowave-to-hdfs.sh --user my-hadoop-user`

NOTE

This only applies to the Accumulo RPM. There is no such requirement for the HBase RPM.

With the exception of the Accumulo RPM mentioned above, there are no restrictions on where you install RPMs. You can install the rest of the RPMs all on a single node for development use or a mix of nodes depending on your cluster configuration.

Running from EMR

For a step by step walkthrough of setting up GeoWave on an EMR cluster, please see our Quickstart Guide.

DataStore Configuration

This section outlines any particular methods for configurations that are necessary for GeoWave-supported datastores.

GeoWave currently supports the following datastores:

- Apache Accumulo Configuration
- Google BigTable Configuration
- Apache HBase Configuration

Apache Accumulo Configuration

Overview

The two high level tasks to configure Accumulo for use with GeoWave are to:

1. Ensure the memory allocations for the master and tablet server processes are adequate
2. Add the GeoWave Accumulo iterator to a classloader. The iterator is a rather large file, so ensure the Accumulo Master process has at least 512m of heap space and the Tablet Server processes have at least 1g of heap space.

The recommended Accumulo configuration for GeoWave requires several manual configuration steps but isolates the GeoWave libraries in application specific classpath(s) reducing the possibility of dependency conflict issues. A single user for all of GeoWave data or a user per data type are two of the many local configuration options. You should just just ensure each namespace containing GeoWave tables is configured to pick up the 'geowave-accumulo.jar'.

Procedure

1. Create a user and namespace.
2. Grant the user ownership permissions on all tables created within the application namespace.
3. Create an application or data set specific classpath.
4. Configure all tables within the namespace to use the application classpath.

```
accumulo shell -u root
createuser geowave ①
createnamespace geowave
grant NameSpace.CREATE_TABLE -ns geowave -u geowave ②
config -s
general.vfs.context.classpath.geowave=hdfs://${MASTER_FQDN}:8020/${ACCUMULO_ROOT}/lib/[^.
].*.jar ③
config -ns geowave -s table.classpath.context=geowave ④
exit
```

① You'll be prompted for a password.

② Ensure the user has ownership of all tables created within the namespace.

③ The Accumulo root path in HDFS varies between hadoop vendors. For Apache and Cloudera it is '/accumulo' and for Hortonworks it is '/apps/accumulo'

④ Link the namespace with the application classpath. Adjust the labels as needed if you've used different user or application names

These manual configuration steps have to be performed before attempting to create GeoWave index tables. After the initial configuration, you may elect to do further user and namespace creation and configuring to provide isolation between groups and data sets.

Managing

After installing a number of different iterators, you may want to figure out which iterators have been configured.

```
# Print all configuration and grep for line containing vfs.context configuration and also
show the following line
accumulo shell -u root -p ROOT_PWD -e "config -np" | grep -A 1
general.vfs.context.classpath
```

You will get back a listing of context classpath override configurations that map the application or user context you configured to a specific iterator jar in HDFS.

Versioning

It's of critical importance to ensure that the various GeoWave components are all the same version and that your client is of the same version that was used to write the data.

Basic

The RPM packaged version of GeoWave puts a timestamp in the name so it's pretty easy to verify that you have a matched set of RPMs installed. After an update of the components, you must restart Accumulo to get vfs to download the new versions and this should keep everything synched.

Compare version and timestamps of installed RPMs

```
[geowaveuser@c1-master ~]$ rpm -qa | grep geowave
geowave-1.0.0-apache-core-1.0.0-201602012009.noarch
geowave-1.0.0-apache-jetty-1.0.0-201602012009.noarch
geowave-1.0.0-apache-accumulo-1.0.0-201602012009.noarch
geowave-1.0.0-apache-tools-1.0.0-201602012009.noarch
```

Advanced

When GeoWave tables are first accessed on a tablet server, the vfs classpath tells Accumulo where to download the jar file from HDFS. The jar file is copied into the local /tmp directory (the default general.vfs.cache.dir setting) and loaded onto the classpath. If there is ever doubt as to if these versions match, you can use the commands below from a tablet server node to verify the version of this artifact.

Commit hash of the jar in HDFS

```
sudo -u hdfs hadoop fs -cat /accumulo/classpath/geowave/geowave-accumulo-build.properties
| grep scm.revision | sed s/project.scm.revision= ①
```

① The root directory of Accumulo in various distributions can vary, so check with 'hadoop fs -ls /' first to ensure you have the correct initial path.

Compare with the versions downloaded locally

```
sudo find /tmp -name "*geowave-accumulo.jar" -exec unzip -p {} build.properties  \; |
grep scm.revision | sed s/project.scm.revision=//
```

Example

```
[spohnae@c1-node-03 ~]$ sudo -u hdfs hadoop fs -cat /${ACCUMULO_ROOT}/lib/geowave-
accumulo-build.properties | grep scm.revision | sed s/project.scm.revision=//
294ffb267e6691de3b9edc80e312bf5af7b2d23f ①
[spohnae@c1-node-03 ~]$ sudo find /tmp -name "*geowave-accumulo.jar" -exec unzip -p {}
build.properties  \; | grep scm.revision | sed s/project.scm.revision=//
294ffb267e6691de3b9edc80e312bf5af7b2d23f ②
294ffb267e6691de3b9edc80e312bf5af7b2d23f ②
25cf0f895bd0318ce4071a4680d6dd85e0b34f6b
```

① This is the version loaded into hdfs and should be present on all tablet servers once Accumulo has been restarted.

② The find command will probably locate a number of different versions depending on how often you clean out /tmp.

There may be multiple versions copies present - one per JVM, the error scenario is when a tablet server is missing the correct iterator jar.

Google BigTable Configuration

There are no additional configuration steps required for Google BigTable, once the RPM has performed the installation.

Apache HBase Configuration

There are no additional configuration steps required for Apache HBase, once the RPM has performed the installation.

Jace JNI Proxies

Using Jace, we are able to create JNI proxy classes for GeoWave that can be used in C/C++ applications.

Boost is required when using the Jace bindings.

Prepackaged Source and Binaries

There is a public GeoWave RPM Repo where you can download a tarball for the GeoWave Jace bindings for your desired platform. If your platform is not available, there is a source tarball which can be used in conjunction with CMake to build the GeoWave Jace bindings for your desired platform.

Generate Proxies and Build from Source

If you want, you can generate and build the Jace proxies yourself. For more details on how to do this, please check out the GeoWave Developer Guide.

Mapnik Plugin Configuration

Mapnik

Mapnik is an open source toolkit for developing mapping applications. GeoWave is supported as a plugin for Mapnik for reading vector data from Accumulo.

PDAL Plugin Configuration

PDAL

The Point Data Abstraction Library PDAL is a BSD licensed library for translating and manipulating point cloud data of various formats. GeoWave is supported as a plugin for PDAL for both reading and writing data to Accumulo.

Note: These instructions assume that you are using prepackaged binaries.

Configure CMake for PDAL

To configure PDAL to run with GeoWave, there are a few CMake options that need to be configured. While some of the options (namely the JAVA options) may configure automatically, some will need to be set manually. Refer to the table below to get an idea for how these options would be configured on Ubuntu 14.04 LTS.

Option	Value	Automatically Configured?
BUILD_PLUGIN_GEOWAVE	ON	
BUILD_GEOWAVE_TESTS	ON	
GEOWAVE_RUNTIME_JAR	/path/to/geowave/geowave-runtime.jar	
GEOWAVE_INCLUDE_DIR	/path/to/geowave/include	
GEOWAVE_LIBRARY	/path/to/geowave/libgeowave.so	
JAVA_AWT_INCLUDE_PATH	/usr/lib/jvm/java-8-oracle/include	X
JAVA_INCLUDE_PATH	/usr/lib/jvm/java-8-oracle/include	X
JAVA_INCLUDE_PATH2	/usr/lib/jvm/java-8-oracle/include/linux	X
JAVA_AWT_LIBRARY	/usr/lib/jvm/java-8-oracle/jre/lib/amd64/libjawt.so	X
JAVA_JVM_LIBRARY	/usr/lib/jvm/java-8-oracle/jre/lib/amd64/server/libjvm.so	X

Note: As Boost is a PDAL dependency, it should already be included.

Build PDAL

Once CMake is configured, you can proceed with your normal PDAL build process.

Last but not least, you should ensure that the libraries specified above are available via *PATH* or *LD_LIBRARY_PATH* when building shared libraries.

Within the PDAL documentation, you can see examples of how GeoWave can be used as both a reader and writer.

Puppet

Overview

A GeoWave Puppet module has been provided as part of both the tar.gz archive bundle and as an RPM. This module can be used to install the various GeoWave services onto separate nodes in a cluster or all onto a single node for development.

There are a couple of different RPM repo settings that may need setting. As the repo is disabled by default to avoid picking up new Accumulo iterator jars without coordinating a service restart, there is likely some customization required for a particular use case. Class parameters are intended to be overridden to provide extensibility.

Options

geowave_version

The desired version of GeoWave to install, ex: '1.0.0'. We support concurrent installs but only one will be active at a time.

hadoop_vendor_version

The Hadoop framework vendor and version against which GeoWave was built. Examples would be cdh5 or hdp2. Check the available packages site for currently supported hadoop distributions.

install_accumulo

Install the GeoWave Accumulo Iterator on this node and upload it into HDFS. This node must have a working HDFS client.

install_app

Install the GeoWave ingest utility on this node. This node must have a working HDFS client.

install_app_server

Install Jetty with Geoserver and GeoWave plugin on this node.

http_port

The port on which the Tomcat application server will run - defaults to 8080.

repo_base_url

Used with the optional geowave::repo class to point the local package management system at a source for GeoWave RPMs. The default location is http://s3.amazonaws.com/geowave-rpms/release/noarch/.

repo_enabled

To pick up an updated Accumulo iterator you'll need to restart the Accumulo service. We don't want

to pick up new RPMs with something like a yum-cron job without coordinating a restart so the repo is disabled by default.

repo_refresh_md

The number of seconds before checking for new RPMs. On a production system the default of every 6 hours should be sufficient, but you can lower this down to 0 for a development system on which you wish to pick up new packages as soon as they are made available.

Examples

Development

Install everything on a one-node development system. Use the GeoWave Development RPM Repo and force a check for new RPMs with every pull (don't use cached metadata).

```
# Dev VM
class { 'geowave::repo':
  repo_enabled    => 1,
  repo_refresh_md => 0,
} ->
class { 'geowave':
  geowave_version       => '1.0.0',
  hadoop_vendor_version => 'apache',
  install_accumulo      => true,
  install_app           => true,
  install_app_server    => true,
}
```

Clustered

Run the application server on a different node. Use a locally maintained rpm repo vs. the one available on the Internet and run the app server on an alternate port, so as not to conflict with another service running on that host.

```
# Master Node
node 'c1-master' {
  class { 'geowave::repo':
    repo_base_url   => 'http://my-local-rpm-repo/geowave-rpms/dev/noarch/',
    repo_enabled    => 1,
  } ->
  class { 'geowave':
    geowave_version       => '1.0.0',
    hadoop_vendor_version => 'apache',
    install_accumulo      => true,
    install_app           => true,
  }
}

# App server node
node 'c1-app-01' {
  class { 'geowave::repo':
    repo_base_url   => 'http://my-local-rpm-repo/geowave-rpms/dev/noarch/',
    repo_enabled    => 1,
  } ->
  class { 'geowave':
    geowave_version       => '1.0.0',
    hadoop_vendor_version => 'apache',
    install_app_server    => true,
    http_port             => '8888',
  }
}
```

Puppet script management

As mentioned in the overview, the scripts are available from within the GeoWave source tar bundle (Search for gz to filter the list). You could also use the RPM package to install and pick up future updates on your puppet server.

Source Archive

Unzip the source archive, locate puppet-scripts.tar.gz, and manage the scripts yourself on your Puppet Server.

RPM

There's a bit of a boostrap issue when first configuring the Puppet server to use the GeoWave puppet RPM as yum won't know about the RPM Repo and the GeoWave Repo Puppet class hasn't been installed yet. There is an RPM available that will set up the yum repo config after which you should install geowave-puppet manually and proceed to configure GeoWave on the rest of the cluster using Puppet.

```
rpm -Uvh http://s3.amazonaws.com/geowave-rpms/release/noarch/geowave-repo-1.0-
3.noarch.rpm
yum --enablerepo=geowave install geowave-puppet
```

Appendices

Version

This documentation was generated for GeoWave version 1.0.0.

GeoWave Security

Datastore Passwords

In order to provide security around account passwords, particularly those entered through command-line, GeoWave is configured to perform encryption on password fields that are configured for datastores or other configured components. To take the topic of passwords even further, GeoWave has also been updated to support multiple options around how to pass in passwords when configuring a new datastore, rather than always having to enter passwords in clear-text at command line.

Password Options

- **pass**: *<password>*
  - This option will allow for a clear-text password to be entered on command-line. It is strongly encouraged not to use this method outside of a local development environment (i.e., NOT in a production environment or where concurrent users are sharing the same system).
- **env**:*<environment variable containing the password>*
  - This option will allow for an environment variable to be used to store the password, and the name of the environment variable to be entered on command-line in place of the password itself.
- **file**:*<path to local file containing the password>*
  - This option will allow for the password to be inside a locally-accessible text file, and the path to file to be entered on command-line in place of the password itself. Please note that the password itself is the ONLY content to be stored in the file as this option will read all content from the file and store that as the password.
- **propfile**:*<path to local properties file containing the password>*:*<property file key to password value>*
  - This option will allow for the password to be stored inside a locally-accessible properties file, and the key that stores the password field to be also specified. The value associated with the specified key will be looked up and stored as the password.
- **stdin**
  - This option will result in the user being prompted after hitting enter, and will prevent the entered value from appearing in terminal history.

NOTE	Users can still continue to enter their password in plain text at command line (just as was done with previous versions of GeoWave), but it is strongly encouraged not to do so outside of a local development environment (i.e., NOT in a production environment or where concurrent users are sharing the same system).

Password Encryption

Passwords are encrypted within GeoWave using a local encryption token key. It is important to not manipulate the key or internal content. By doing so, you are compromising the likelihood of not being able to encrypt new data or decrypt already encrypted data.

NOTE

It is assumed that a *geowave* system command alias is registered in the terminal session being run through.

To test this, type 'geowave' (no quotes) and press Enter.

If a list of GeoWave options is returned, then the system command alias is available. Otherwise, if an error similar to or containing the term *'geowave: command not found'* is returned, the system command alias has not been registered. For details on how to register the 'geowave' system command alias, please refer to the GeoWave Developer Guide.

If the alias system command is not registered, the full java command - e.g. java -cp {GEOWAVE_HOME} {GEOWAVE_JAR}) will need to be used in place of the 'geowave' alias.

In the event that the encryption token key is compromised, or thought to be compromised, a new token key can very easily be generated using a GeoWave system command.

```
$ geowave config newcryptokey
```

The above command will re-encrypt all passwords already configured against the new token key. As a result, the previous token key is obselete and can no longer be used.

NOTE

This option is only useful to counter the event that only the token key file is compromised. In the event that both the token key file and encrypted password value have been compromised, it is recommended that steps are taken to change the datastore password and re-configure GeoWave to use the new password.

Configuring Console Echo

When the 'stdin' option is specified for passwords to be entered at command line, it is recognized that there are circumstances where the console echo is wanted to be enabled (i.e., someone looking over your shoulder), and other times where the console echo is wanted to be disabled.

For configuring the default console echo setting:

```
$ geowave config set geowave.console.default.echo.enabled={true|false}
```

The above command will set the default setting for all console prompts. Default is false if not specified, meaning any characters that are typed (when console echo is disabled) are not shown on the screen.

GeoWave provides the ability to override the console echo setting for passwords specifically. For configuring the password console echo setting:

```
$ geowave config set geowave.console.password.echo.enabled={true|false}
```

If the above is specified, this setting will be applied for passwords when a user is promoted for input. By default, if the passwords console echo is not specified, the system will use the console default echo setting.

Enabling/Disabling Password Encryption

GeoWave provides the ability to enable or disable password encryption as it is seen necessary. By default, password encryption is enabled, but can be disabled for debugging purposes. For configuring the password encryption enabled setting:

```
$ geowave config set geowave.encryption.enabled={true|false}
```

NOTE	It is HIGHLY discouraged against disabling password encryption, particularly in a production (or similar) environment. While this option is available for assisting with debugging credentials, it should be avoided outside of production-like environments in order to avoid leaking data source credentials to unauthorized parties.