pitney bowes

Spectrum™ Routing for Big Data

Version 3.0.0

User Guide

Table of Contents

1 - Welcome

In this section

Introducing Spectrum™ Routing for Big Data

Spectrum™ Routing for Big Data is a toolkit that can be used for processing enterprise data for large-scale spatial analysis. Billions of records from a single file can be processed in parallel using MapReduce, Hive, and Apache Spark's cluster processing framework, yielding results faster than ever. Unlike the traditional processing techniques that took weeks to process data, using this product, data processing can now be done in a just a few hours.

Spectrum™ Routing for Big Data Architecture

What is Spectrum™ Routing for Big Data?

The Spectrum™ Routing for Big Data packages Routing components into an SDK for Big Data platforms like Hadoop for MapReduce and Spark, Hive etc.

SDK provides:

- Integration APIs for (PointToPointRoute, IsoChrone and IsoDistance)
- Input datasets and metadata

API Types:

- Pre-built Hive UDF wrappers for Routing operations
- Core Routing APIs (Security enabled via Kerberos and Apache Sentry for Hive)

System Requirements and Dependencies

Spectrum™ Routing for Big Data is collection of jar files that can be deployed to your Hadoop system.

This product is verified on the following Hadoop Distributions.

- Cloudera 5.12
- Hortonworks 2.6
- EMR 5.10

To use these jar files, you must be familiar with configuring Hadoop in Hortonworks, Cloudera, or EMR and developing applications for distributed processing. For more information, refer to **Hortonworks**, **Cloudera**, or **EMR** documentation.

To use the product, the following must be installed on your system:

for MapReduce, Spark and Zeppelin Notebook:

- Java JDK version 1.7 or above
- Hadoop version 2.6.0 or above
- Spark version 1.6.0 or above
- Zeppelin Notebook not supported in Cloudera

for Hive:

- Hive version 1.2.1 or above

for Hive Client

- For example, Beeline

2 - Routing UDFs

In this section

Overview

This section describes the Hive user-defined functions (UDFs) available for performing routing operations.

- **PointToPointRoute:** Helps generate the routing information required to travel between two distinct points.
- **IsoChrone:** Helps find points that can be reached from the specified starting location in the specified duration.
- **IsoDistance:** Helps find points that lie at some specified distance from the given starting location, considering every possible route.

Working with Routing Hive UDFs

Setting Up Routing Hive UDFs

This section takes you through the steps you need to perform to setup the Routing Hive UDFs:

1. Log on to Name Node using an FTP tool, for example, WinSCP.
2. Copy the `spectrum-bigdata-routing-version.zip` file to the home directory.
3. Use the following command to create a new folder, `PB`, in the root folder:

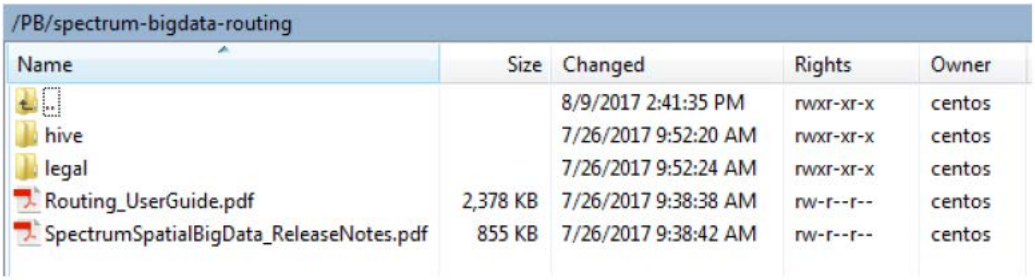
```
sudo mkdir /PB
```

4. Use the following code to grant access rights to the new folder and all its sub-folders to all other logged-in users.

```
sudo chown -R centos:centos /PB
```

5. Unzip the file `spectrum-bigdata-routing-version.zip` and extract its contents in the new folder, `PB`, as shown below:

```
unzip spectrum-bigdata-routing-version.zip -d
/PB/spectrum-bigdata-routing
```



Setting Up Routing Data

Routing Hive UDFs are required in order to access a Routing dataset during its operation.

Note the location of the `dbList.json` configuration file; it contains details about the Routing datasets that are installed.

To know how to setup a Routing dataset, refer to the *Data Installation* section in the *Global Routing API User Guide*.

Running the Routing Hive Job from Command-Line

This section takes you through the steps you need to perform to run the Routing Hive job from the command-line.

1. Use the following command to start the Hive shell:

```
$ hive
```

   Alternatively, use the following command to start Hive in INFO mode:

```
$ hive --hiveconf hive.root.logger=INFO,console
```

2. Use the following command to add the Routing JAR to Hive:

```
add jar
/dir/on/server/spectrum-bigdata-spatial-routing-hive-version.jar;
```

3. Set the mandatory and optional system properties as needed. The `dbConfigJSONFile` system property is a mandatory property which specifies the location of the cofiguration file. Routing Hive UDFs require the following properties to be set in order to start a GRA engine with the given configurations:

   - **dbConfigJSONFile** (Mandatory): Specifies the local file system path of the `dblist.json` file. Use the following command in the Hive console to set this property:

```
set
system:dbConfigJSONFile=<path_to_folder_containing_dblist.json>/dblist.json;
```

   - **routeTimeout** (Optional): Specifies the amount of time, in milliseconds, allowed for a point-to-point routing request to be processed and completed. If the request is not processed or completed within the specified duration, it is terminated. Use the following command in the Hive console to set this property:

```
set system:routeTimeout=10000;
```

   - **routeDefaultCoordSys** (Optional): Specifies the coordinate system to be used by default. Use the following command in the Hive console to set this property:

```
set system:routeDefaultCoordSys="EPSG:4326";
```

   - **isoTimeout** (Optional): Specifies the amount of time (in milliseconds) allowed for a boundary routing request to be processed and completed. If the request is not processesed or completed

within the specified duration, it is terminated. Use the following command in the Hive console to set this property:

```
set system:isoTimeout=17000;
```

- **allowFallback** (Optional): This is a Boolean property that controls whether the engine can use major roads in case the other roads cannot be used. If set to `true`, the engine can use the major roads. Use the following command in the Hive console to set this property:

```
set system:allowFallback=true;
```

- **shortProcessThreads** (Optional): Specifies the number of threads to be used for a point-to-point route requests and routing data requests. Use the following command in the Hive console to set this property:

```
set system:shortProcessThreads=16;
```

- **longProcessThreads** (Optional): Specifies the number of threads to be used for isochrone requests. Use the following command in the Hive console to set this property:

```
set system:longProcessThreads=16;
```

- **graEngineTimeoutInSeconnds** (Optional): Specifies the duration (in seconds) in which a GRA engine is disposed if it does not receive any request. The default value of this property is -1, which means that the GRA engines will never time out. Use the following command in the Hive console to set this property:

```
set system:graEngineTimeoutInSeconnds=10;
```

If you do not want the GRA engines to be disposed, setting this property is not required. If you have already specified a value for this property, but later you do not want the GRA engines to be disposed, leave the property blank or set it to -1.

```
set system:graEngineTimeoutInSeconnds=;
                                      set
system:graEngineTimeoutInSeconnds=-1
```

4. Register the Routing UDFs using the commands given below:

```
hive> create temporary function PointToPointRoute as
'com.pb.bigdata.spatial.routing.hive.PointToPointRoute';
```

```
hive> create temporary function IsoChrone as
'com.pb.bigdata.spatial.routing.hive.IsoChrone';
```

```
hive> create temporary function IsoDistance as
'com.pb.bigdata.spatial.routing.hive.IsoDistance';
```

The Routing functions are now ready to use.

For more information, see the **PointToPointRoute** on page 12, **IsoChrone** on page 14, or **IsoDistance** on page 15.

PointToPointRoute

Syntax

```
Select PointToPointRoute(X_Column_Of_Start_Point,
Y_Column_Of_Start_Point,
Start_Point_CRS, X_Column_Of_End_Point, Y_Column_Of_End_Point,
End_Point_CRS) From Table_Name;
```

Description

The `PointToPointRoute` UDF helps extract the routing information required for traveling between two distinct points. It takes the starting location, the ending location, the distance (this is optional) and the time units, and it returns the total distance and duration of a route that is either the fastest or the shortest.

Parameters

X_Column_Of_Start_Point	The X coordinate of the starting location.
Y_Column_Of_Start_Point	The Y coordinate of the starting location
Start_Point_CRS	The coordinate reference system used for the starting location.
End_Point_CRS	The coordinate reference system used for the ending location.

X_Column_Of_End_Point	The X coordinate of the ending location.
Y_Column_Of_End_Point	The Y coordinate of the ending location.

Return Values

Depending on the query, this function returns values described in the table below:

Time	The shortest duration of traveling between two points in minutes.
Distance	The total distance of the shortest path between two points in meters.

Note: To know more about the supported distance and time units, refer to the *Pitney Bowes Global Routing API SDK Product Guide*.

Example

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
 42.677640, "epsg:4326" );
```

This function call returns the distance of the fastest route between two points in meters and the total travel duration in minutes.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
 42.677640, "epsg:4326" ).distance;
```

This function call returns the distance of the shortest route between two points in meters.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
 42.677640, "epsg:4326" ).time;
```

This function call returns the duration of the fastest route between two points in minutes.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
 42.677640, "epsg:4326", "ft", null );
```

This function call returns the distance of the fastest route between two points in feet and the total travel duration in minutes.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
 42.677640, "epsg:4326", null, "msec" );
```

This function call returns the distance of the fastest route between two points in feet and the total travel duration milliseconds.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
 42.677640, "epsg:4326", null, null );
```

This function call returns the distance of the fastest route between two points in meters and the total travel duration in minutes.

The Routing Hive UDFs are interoperable with the Spatial and Geocoding Hive UDFs, as shown in the example given below:

```
select PointToPointRoute(Geocode("485A","Watervliet Shaker
Rd","","Latham","NY","12110","usa").geometry, Geocode("204-240","Jordan
 Rd","","Troy","NY","12180","usa").geometry);
```

> **Note:** See the *Spectrum™ Geocoding for Big Data User Guide* and *Spectrum™ Location Intelligence for Big Data User Guide* for more information on how to use the Geocoding and Spatial UDFs.

IsoChrone

Syntax

```
Select IsoChrone(X_Column, Y_Column, Coordinate_Reference_System,
Cost_Time, Time_Unit) From Table_Name;
```

Description

An isochrone is a polygon connecting points that take the same amount of time to reach from a particular point. This function returns an isochrone connecting a set of points that can be reached from the specified starting location in the given duration.

The `IsoChrone` UDF uses the `GetTravelBoundary` operation of the Global Routing API to generate an IsoChrone. The `GetTravelBoundary` operation helps determine the drive or walk time or the distance boundary from a given location. This function generates polygons corresponding to an isochrone or isodistance calculation.

Parameters

X_Column	Specifies the X coordinate of the starting location.
Y_Column	Specifies the Y coordinate of the starting location.
Coordinate_Reference_System	Specifies the coordinate reference system to be used.

Cost_Time	Specifies the time available to reach the end point.
Time_Unit	Specifies the unit of time.

Return Value

Polygon Geometry	The `WritableGeometry` of an isochrone.

Example

```
Select IsoChrone("-77.088217",38.937072,"epsg:4326",3,"min");
```

```
Select IsoChrone(Points.X, Points.Y,3,"min") From Points;
```

Both these queries return a polygon geometry comprising all the points that can be reached from the specified starting point in the given time, that is, three minutes.

Furthermore, you can use the `WritableGeometry` returned by the IsoChrone UDF with the Spatial UDFs for further processing, as shown in the examples given below:

```
Select ToWKT(IsoChrone(true,38.937072,"epsg:4326",3,"min"));
```

```
select Within(ST_POINT(-77.088217,
38.937072)),IsoChrone("-77.088217",38.937072,"epsg:4326",3,"min"));
```

IsoDistance

Syntax

```
Select IsoDistance(X_Column, Y_Column,  Coordinate_Reference_System,
Cost_Distance,  Linear_Unit) From Table_Name;
```

Description

An Isodistance is a polygon representing specific distance intervals from a particular point extended out along all possible paths. This function returns an isodistance connecting a set of points that lie at a specified distance from the given starting point, considering every possible route.

The `IsoDistance` UDF uses the `GetTravelBoundary` operation of the Global Routing API to generate the Isodistance. The `GetTravelBoundary` operations helps determine the drive or walk time or the distance boundary from a given location. This function generates polygons corresponding to an isochrone or isodistance calculation.

Parameters

X_Column	Specifies the X coordinate of the starting location.
Y_Column	Specifies the Y coordinate of the starting location.
Coordinate_Reference_System	Specifies the coordinate reference system to be used.
Cost_Distance	Specifies the distance to be traveled from the available to reach the end point.
Linear_Unit	Specifies the unit of distance.

Return Values

Polygon Geometry	Returns the `WritableGeometry` of an isodistance.

Note: For values of distance linear units, refer to the *Pitney Bowes Global Routing API SDK* product guide.

Example

```
Select IsoDistance (-77.088217, 38.937072, "epsg:4326", 3, "km");
```

```
Select IsoDistance (Points.X, Points.Y, 3, "km") From Points;
```

Both these queries return a polygon geometry comprising all the points that lie at the specified distance from the starting point.

You can use the `WritableGeometry` returned by IsoDistance with the Spatial UDFs for further processing, as shown below:

```
Select ToWKT(IsoDistance (-77.088217, 38.937072, "epsg:4326", 3, "km"));
```

```
Select Within(ST_POINT(-77.088217, 38.937072), IsoDistance (-77.088217,
 38.937072, "epsg:4326", 3, "km"));
```

Copyright

3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com