precisely

Spectrum Spatial for Big Data

Version 4.1

Release Notes

This document contains the new and updated features for Spectrum Spatial for Big Data.

Contents:

What's New?

The Spectrum Spatial for Big Data version 4.1 suite of products contains the following new and updated features. For more information on all these updates, see the Spectrum Spatial for Big Data user guides and content on the **Spectrum Spatial for Big Data** documentation landing page.

Spectrum Geocoding for Big Data

- A new sample **Geocoding Application for Kubernetes** is now available on **GitHub**. This sample demonstrates the deployment of the Spectrum Global Geocoding API in a cloud-native environment. It provides elastic REST endpoints for geocoding, reverse geocoding, interactive geocoding, and key lookup that scale based on the number of active connections.
- New code samples that demonstrate some of the capabilities of Spectrum Geocoding for Big Data are now available on **GitHub**:
  - **Geohash Aggregation** - Demonstrates how to return geohashes for point locations and aggregate the data in each geohash. The locations in this sample come from social service requests into the NYC 311 information system. Each request contains a NYC agency responsible for handling the request. The sample computes a geohash value for each location at a geohash precision of 7. The data is then grouped by a common geohash and counted.
  - **Multipass Geocoding** - This sample for the Spark Geocoding API in Scala demonstrates how to improve geocoding results by performing multipass geocoding. With multipass geocoding, for all first-pass results without point-level precision, a second geocoding pass is run using single line address matching, which may return more accurate geocodes.
- Configuring your geocoding datasets is now easier and done automatically; it is no longer necessary to use the CLI. To set up geocoding, point to either a single dataset (extracted or an unextracted SPD), or a directory of datasets using the new data file path variable/property listed below. One or more file paths can be specified. In addition, you can optionally set the extraction location for the geocoding datasets using the new extraction location variable/property; if not specified, the default location is the same directory as the SPD. The new Hive variables and Spark driver properties for the data and SPD extraction locations are as follows:

Hive:	`pb.geocoding.data.location`	`pb.geocoding.extraction.location`
Spark driver:	`--geocoding-data-location`	`--extraction-location`

  For more information, see the *Spectrum Geocoding for Big Data Installation Guide* for your platform.
- Support has been added for the Parquet format for geocoding and reverse geocoding Spark jobs. New options have been added to specify the Parquet format for either or both input and output files, as well as setting Parquet options such as compression method.
- A new `--limit` option is now available for geocoding and reverse geocoding Spark jobs, which allows you to set the maximum number of records to be processed.

- The Spectrum Global Geocoding API can now return the GEOHASH attribute using a POST request. GeoHash uses the latitude and longitude of the geocoded candidate to return a 32-bit encoded string representing an area. The more characters in the string, the more precise the location. By default, the precision level is set to the highest precision of 12 characters. For more information about the GeoHash feature, see the ***Spectrum Global Geocoding SDK Developer Guide*** and ***Spectrum Global Geocoding SDK Release Notes v3.1.71***.

Spectrum Location Intelligence for Big Data

- New code samples that demonstrate some of the capabilities of the Spectrum Location Intelligence for Big Data capabilities are now available on **GitHub**:
  - **Enrichment Sample using Boundaries** - Demonstrates how to use Spark to enrich a CSV containing point data with attributes from a spatial file based on a point in polygon search. Enriching your point data with another dataset can provide you with additional context. For example, let's say that you have a list of clients represented by the Address Fabric sample. Understanding what risks are associated with those clients can help you price products such as insurance. This sample demonstrates joining the Address Fabric to the Crime Index data.
  - **Point Enrichment** - Demonstrates how to use Spark to enrich a CSV containing point data with attributes associated with points within some max distance. Enriching your point data with another dataset can provide you with additional context. For example, let's say that you have a list of customers represented by the Address Fabric sample, understanding which businesses are nearby to each of them can help determine where they might be shopping. This sample demonstrates joining the Address Fabric to the Point of Interest (POI) dataset with a distance of 0.5 miles.
- The `JoinByDistance` method enriches point data with attributes associated with points within some maximum distance. There are now two new options to limit the returned match results: The `LimitMatches` option limits the number of joined results for each source dataframe record; the `LimitMethod` option supports limiting the matches returned by row number, rank of rows with gaps, or dense rank (without any gaps).
- Added support for ESRI geodatabase spatial file format. The Geodatabase API requires ESRI ArcGIS geodatabase version 10 or above. For API usage, see the Scaladocs in the distribution ZIP.
- New APIs have been added in Spark to better facilitate spatial searches against file-based tables such as TAB, shapefile, and geodatabase. For API usage, see the Scaladocs in the distribution ZIP. For further information and example usage, see the geodatabase subproject in the Enrichment sample on **GitHub**.

For more information, see the *Location Intelligence for Big Data User Guide* on the **Spectrum Spatial for Big Data** documentation landing page, as well as the Scaladocs available in your distribution:

`/pb/li/software/spark2/sdk/scaladocs/bigdata`

Breaking Changes

For Spark, separate jar files are now provided that are specific to Scala version support. The following paths and filenames need to be used when installing the jar, executing Spark job commands, and using the jar in Zeppelin or Hue notebooks (for Location Intelligence and Routing).

- **Spectrum Geocoding for Big Data**

  Note: The Geocoding SDK does not use Scala directly, so no Scala-specific library is required. However, the provided driver application will require this change.

  Scala 2.12:

  ```
  /pb/geocoding/software/spark2/driver_2.12/spectrum-bigdata-geocoding-spark2_2.12-4.1-all.jar
  ```

  Scala 2.11:

  ```
  /pb/geocoding/software/spark2/driver_2.11/spectrum-bigdata-geocoding-spark2_2.11-4.1-all.jar
  ```

- **Spectrum Location Intelligence for Big Data**

  Scala 2.12:

  ```
  /pb/li/software/spark2/sdk/lib/spectrum-bigdata-li-sdk-spark2_2.12-4.1.jar
  ```

  Scala 2.11:

  ```
  /pb/li/software/spark2/sdk/lib/spectrum-bigdata-li-sdk-spark2_2.11-4.1.jar
  ```

- **Spectrum Routing for Big Data**

  Scala 2.12:

  ```
  /pb/routing/software/spark2/sdk/lib/spectrum-bigdata-routing-sdk-spark2_2.12-4.1.jar
  ```

  Scala 2.11:

  ```
  /pb/routing/software/spark2/sdk/lib/spectrum-bigdata-routing-sdk-spark2_2.11-4.1.jar
  ```

Hadoop Distribution and Framework Support

New

- Added support for EMR 5.30 and 6.0
- Added support for Scala 2.12

Library Upgrades:

- Spectrum Global Geocoding SDK (GGS) 3.1.84
- Global Routing SDK (GRA) 2.0.12
- Location Intelligence SDK (LI SDK) 1.13.7

Known Issues

- (HAD-4323) Some types of queries will cause Hive to evaluate UDFs in the HiveServer2 process space instead of on a data node. The Routing UDFs in particular use a significant amount of memory and can shut down the Hive server due to memory constraints. To process these queries, we recommend increasing the amount of memory available to the HiveServer2 process (for example, by setting HADOOP_HEAPSIZE in hive-env.sh).
- (HAD-2967) A null pointer exception is thrown while running the following constant geocoding queries with Hive on Spark or TEZ:

```
SELECT ReverseGeocode(-76.97921145819674,38.89325106109181,"epsg:4326");
```

```
SELECT Geocode("","One Second Street","","Jersey City","","07302","USA");
```

This is due to an issue in Hive: **https://issues.apache.org/jira/browse/HIVE-16587**

Workaround

```
CREATE table dual (country string);
INSERT into dual values("usa");
SELECT Geocode('','2920 LANGSTON PLACE SOUTHEAST APARTMENT 101',
'','WASHINGTON','DC','20020',a.country) from dual a;
```

System Requirements and Dependencies

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

This product is verified on the following Hadoop distributions.

- Cloudera 5.12, 6.2
- Hortonworks 3.0, 3.1
- EMR 5.30, 6.0

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 Hive:

- Hive version 1.2.1 or above
- Hive in Spectrum Geocoding for Big Data is not supported on Cloudera 5.12

for Hive Client

- Beeline, for example

for Spark and Zeppelin Notebook:

- Java JDK version 1.8 or above
- Hadoop version 2.6.0 or above
- Spark version 2.0 or above
- Zeppelin Notebook is not supported in Cloudera

GGS Memory Requirements

- The amount of memory required depends on the deployment scenario implemented. The basic memory requirement for the Spectrum Global Geocoding SDK is 16 GB RAM. We recommend 32 GB RAM.
- If you are using a large number of datasets (more than 20), review your **minimum heap size** setting. Consider increasing it to at least 8 GB to prevent out of memory exception errors.
- For more information, see the ***Spectrum Global Geocoding SDK Developer Guide***.

precisely

2 Blue Hill Plaza, Fl 3 #1563
Pearl River, NY 10965-3113
USA

www.precisely.com