Geophysical Research Abstracts
Vol. 19, EGU2017-10205-2, 2017
EGU General Assembly 2017

Ibmdbpy-spatial : An Open-source implementation of in-database geospatial analytics in Python

Avipsa Roy (1,2), Edouard Fouché (2), Rafael Rodriguez Morales (2), and Gregor Moehler (2)

(1) Institute for Geoinformatics, University of Münster, Münster, Germany (a_roy001@uni-muenster.de), (2) IBM Research and Development GmbH, Böblingen, Germany

As the amount of spatial data acquired from several geodetic sources has grown over the years and as data infrastructure has become more powerful, the need for adoption of in-database analytic technology within geosciences has grown rapidly. In-database analytics on spatial data stored in a traditional enterprise data warehouse enables much faster retrieval and analysis for making better predictions about risks and opportunities, identifying trends and spot anomalies. Although there are a number of open-source spatial analysis libraries like *geopandas* and *shapely* available today, most of them have been restricted to manipulation and analysis of geometric objects with a dependency on GEOS and similar libraries. We present an open-source software package, written in Python, to fill the gap between spatial analysis and in-database analytics. *Ibmdbpy-spatial* provides a geospatial extension to the *ibmdbpy* package, implemented in 2015. It provides an interface for spatial data manipulation and access to in-database algorithms in *IBM dashDB*, a data warehouse platform with a spatial extender that runs as a service on IBM's cloud platform called *Bluemix*.

Working in-database reduces the network overload, as the complete data need not be replicated into the user's local system altogether and only a subset of the entire dataset can be fetched into memory in a single instance. *Ibmdbpy-spatial* accelerates Python analytics by seamlessly pushing operations written in Python into the underlying database for execution using the *dashDB* spatial extender, thereby benefiting from in-database performance-enhancing features, such as columnar storage and parallel processing. The package is currently supported on Python versions from 2.7 up to 3.4.

The basic architecture of the package consists of three main components - 1) a connection to the dashDB represented by the instance *IdaDataBase*, which uses a middleware API namely - *pypyodbc* or *jaydebeapi* to establish the database connection via ODBC or JDBC respectively, 2) an instance to represent the spatial data stored in the database as a dataframe in Python, called the *IdaGeoDataFrame*, with a specific geometry attribute which recognises a planar geometry column in *dashDB* and 3) Python wrappers for spatial functions like *within, distance, area, buffer* and more which *dashDB* currently supports to make the querying process from Python much simpler for the users. The spatial functions translate well-known *geopandas*-like syntax into SQL queries utilising the database connection to perform spatial operations in-database and can operate on single geometries as well two different geometries from different IdaGeoDataFrames. The in-database queries strictly follow the standards of OpenGIS Implementation Specification for Geographic information - Simple feature access for SQL.

The results of the operations obtained can thereby be accessed dynamically via interactive Jupyter notebooks from any system which supports Python, without any additional dependencies and can also be combined with other open source libraries such as *matplotlib* and *folium* in-built within Jupyter notebooks for visualization purposes. We built a use case to analyse crime hotspots in New York city to validate our implementation and visualized the results as a choropleth map for each borough.