PostGIS ♡ Protobuf

Introduction

- Björn Harrtell, Sweden
- Septima, Denmark

Note:
At Septima I work as a geospatial developer.

Subject

- Encoding Protobuf formats in PostGIS
- Mapbox Vector Tiles (lossy)
- Geobuf (lossless)

Note:
This talk is about a upcoming feature in PostGIS 2.4 to support outputting data in two Protobuf based formats, Mapbox Vector Tiles and Geobuf. It's important to note that vector tiles is a lossy format where as Geobuf is lossless.

Protobuf is a binary encoding specification from Google for structured data and I think that it can perhaps be seen as a binary equivalent of XML.

Why encode vector tiles in PostGIS?

- Custom projections
- Simplify toolchain
- Reduce I/O for potential performance wins

Note:
A few years back I had an assigment to build a mobile client with offline support and the ability to download and handle many layers of vector data for which vector tiles seemed the best available choice because it's optimized for rendering and size. A requirement was to use a swedish reference system.

So my main reason to look into the details of encoding vector tiles was to support custom projections.

I did hack my own toolchain in Java at one point and while it worked I wasn't happy to maintain it. So I was interested in a more simple toolchain.

It bothered me that toolchains was essentially middleware requiring I/O and serialization of the source data, so I thought why not do this directly in PostGIS to simplify and reduce I/O.

Two new functions for vector tiles

- ST_AsMVTGeom (https://postgis.net/docs/manual-dev/ST_AsMVTGeom.html)
- ST_AsMVT (https://postgis.net/docs/manual-dev/ST_AsMVT.html)

Note:
The two new functions for vector tiles are ST_AsMVT and ST_ASMVTGeom.

- Show the docs, show the vector tile specification

ST_AsMVTGeom is used to transform a geometry into tile coordinate space.

ST_AsMVT is an aggregate function to encode a set of rows with transformed geometry + attributes into the binary vector tiles format.

The initial version of ST_AsMVT took a SQL string as input and queried it internally. It was Paul's suggestion to make it an aggregate function which required some deep diving into Postgres internals.

The initial implementation was only a single function, ST_AsMVT, but at some point Blake Thompson from Mapbox informed me that the vector tile specification require geometry to be OGC valid and even if your input geometry is valid it can become invalid when transformed to the local coordinate space of a vector tile. So I reworked the implementation and split it into two functions to make it explicit that there is a geometry transformation step. This has the additional benefit that ST_AsMVTGeom should be able to take advantage of the new Postgres parallel capabilities, though I have not tested this myself.

I use PostGIS functions that in turn uses GEOS for ST_AsMVTGeom to clip and correct geometries. Mapbox has their own library https://github.com/mapbox/wagyu with similar functionality. I think it would be interesting to integrate this into PostGIS at some point with the potential of being quite a bit faster.

Function for Geobuf

- ST_AsGeobuf (https://postgis.net/docs/manual-dev/ST_AsGeobuf.html)
- Similar to GeoJSON but more compact

Note:
I've also implemented ST_AsGeobuf, which I actually did before vector tiles because while also based on protobuf it's a more simple encoding so was a good start for me.

Can be a useful alternative to GeoJSON when you need larger volumes of data on the client side. Can complement vector tiles for when you need get the lossless source data.

Demonstration

- Basic usage

mvtile.sql

```
SELECT ST_AsMVT('land', 512, 'geom', q)
FROM (SELECT
  id,
  ST_AsMVTGeom(
    geometry,

ST_MakeEnvelope(:xmin,:ymin,:xmax,:ymax,3857),
    512, 0, true
  ) AS geom
FROM foss4g.ne_50m_land_3857
WHERE
  geometry &&
ST_MakeEnvelope(:xmin,:ymin,:xmax,:ymax,3857)
) as q WHERE geom IS NOT NULL
```

psql

```
psql -At \
  -v xmin=-20037508.342789244 \
  -v ymin=0 \
  -v xmax=0 \
  -v ymax=20037508.342789244 \
  -f mvtile.sql | xxd -r -p >./1/0/0.pbf
```

Note:
This is what I used to produce the examples shown.

For the SQL part we have ST_AsMVT with 'land' as layer name, extent 512 which is the "resolution" of the vector tile, the name of the geometry column in the input rows and then the rows to aggregate.

Note that we're requiring that geometry is not null, which can happen due to small geometries collapsing because they are smaller than a tile coordinate pixel.

The subquery filters input geometry on a bbox which should be the geometric bounds of the tile. If we where to use a tile buffer (to fix boundary render issues) we would need to add that to this bbox.

Then we have the source table foss4g.ne_50m_land_3857 and the call to ST_AsMVTGeom. ST_AsMVTGeom also needs the geometric bounds of the tile but should not include any buffer. Again we have to specify the resolution of the tile (512), a buffer of 0 and last parameter is true to have it clip the geometries on the tile boundary + buffer.

Now I can use mvtile.sql with psql to query for the first quadrant tile of the world.

Note that psql does not output raw binary, it's in hex encoded format so I use the xxd command to decode it to raw binary.

Results

- Single tile (https://bjornharrtell.github.io/presentations/vectortiles/e
- Four tiles with OL 4.0 (https://bjornharrtell.github.io/presentations/vectortiles/e
- Four tiles with OL 4.2 (https://bjornharrtell.github.io/presentations/vectortiles/e
- Higher resolution (https://bjornharrtell.github.io/presentations/vectortiles/e
- Random fill color (https://bjornharrtell.github.io/presentations/vectortiles/e

Postgres details

- Why not COPY?
- Does not output raw binary
- "*The binary file format consists of a file header, zero or more tuples containing the row data, and a file trailer. Headers and data are in network byte order.*"
- Use a PG driver and read the binary directly for optimal use

Note:
You might ask why I'm not using the COPY command and it's because COPY does not output raw binary, it has headers.

If there was an option to output raw binary I could simply output the tiles directy. Perhaps an opportunity for a future contribution?

So, the current optimal way to use ST_AsMVT is through a Postgres driver where you can read the binary directly.

How?

- protobuf-c (https://github.com/protobuf-c/protobuf-c)
- vector-tile-spec (https://github.com/mapbox/vector-tile-spec/tree/master/2.1)
- Pitch the idea and a POC on the postgis-devel list to get interest
- Talk Paul Ramsey out of having to make the code unit testable (!)
- Get sponsoring from Carto
- Regression tests

Note:
Arguments against unit testable code was probably pretty weak but at least Paul
did not say it was unacceptable. :)

Future development

- Investigate use of https://github.com/mapbox/wagyu
- Encourage use of ST_AsMVT in tile servers like t-rex

End!

@bjornharrtell (https://twitter.com/bjornharrtell)