Cloud optimized formats for rasters and vectors explained

Pirmin Kalberer
@implgeo
Sourcepole, Zurich
www.sourcepole.com

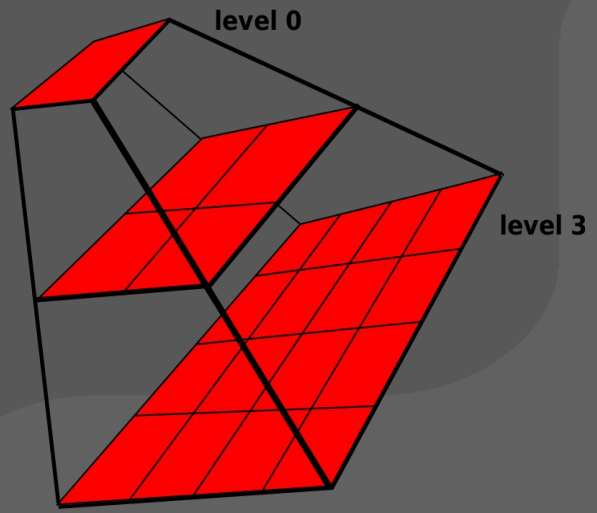
SOURCEPOLE
Linux & Open Source Solutions

Web access - what's the problem?

- Geospatial data files can be very big
- Reading files over a network much slower than from local disc
- Loading the whole file may be unfeasible

Solution #1: Tile caches

- Split the file into tiles
  → Reading only the required files can be much faster
- Works well for raster data (XYZ, WMTS)
- Disadvantage: Tile caches are expensive



Solution #2: Sort your file content

- Optimize order of file content for reading chunks
- Use HTTP Range requests for partial reading
- Find a catchy name → “cloud optimized”

- HTTP Range request:

```
curl http://example.com/image.tif -H "Range: bytes=0-1023"
```

```
GET /image.tif HTTP/1.1
Host: example.com
Range: bytes=0-1023
```

Exhibit #1: TIFF/GeoTIFF

GeoTIFF:

- Supports tiles
- Supports overviews

Cloud Optimized GeoTIFF (COG):

- Ordered Metadata
- Ordered imagery
- https://www.cogeo.org/

Creating Cloud Optimized GeoTIFFs

- **GeoTIFF to COG:**

```
gdal_translate in.tif cog.tif -co TILED=YES
-co COPY_SRC_OVERVIEWS=YES -co COMPRESS=DEFLATE
```

- **Since GDAL 3.1:**

```
gdal_translate in.tif cog.tif -of COG -co COMPRESS=DEFLATE
```

Reading Cloud Optimized GeoTIFFs

- **GDAL datasource (QGIS, Mapserver, etc.):**
- `/vsicurl/https://s3-us-west-2.amazonaws.com/planet-disaster-data/hurricane-harvey/SkySat_Freeport_s03_20170831T162740Z3.tif`

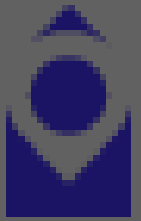
- **In the browser with geotiff.js**
  - https://geotiffjs.github.io/
- **Leaflet: georaster-layer-for-leaflet**
  - https://github.com/geotiff/georaster-layer-for-leaflet

Exhibit #2: Vector data

- GeoJSON/GML: forget it
- GPKG: Optimized for disk block IO
- Shapfiles: Not bad, but too limited and too many sidecars

FlatGeobuf:

- Metadata with index (packed Hilbert R-Tree)
- Ordered vector data
- Based on Flatbuffers (schema, portable, verification)
- https://flatgeobuf.org/



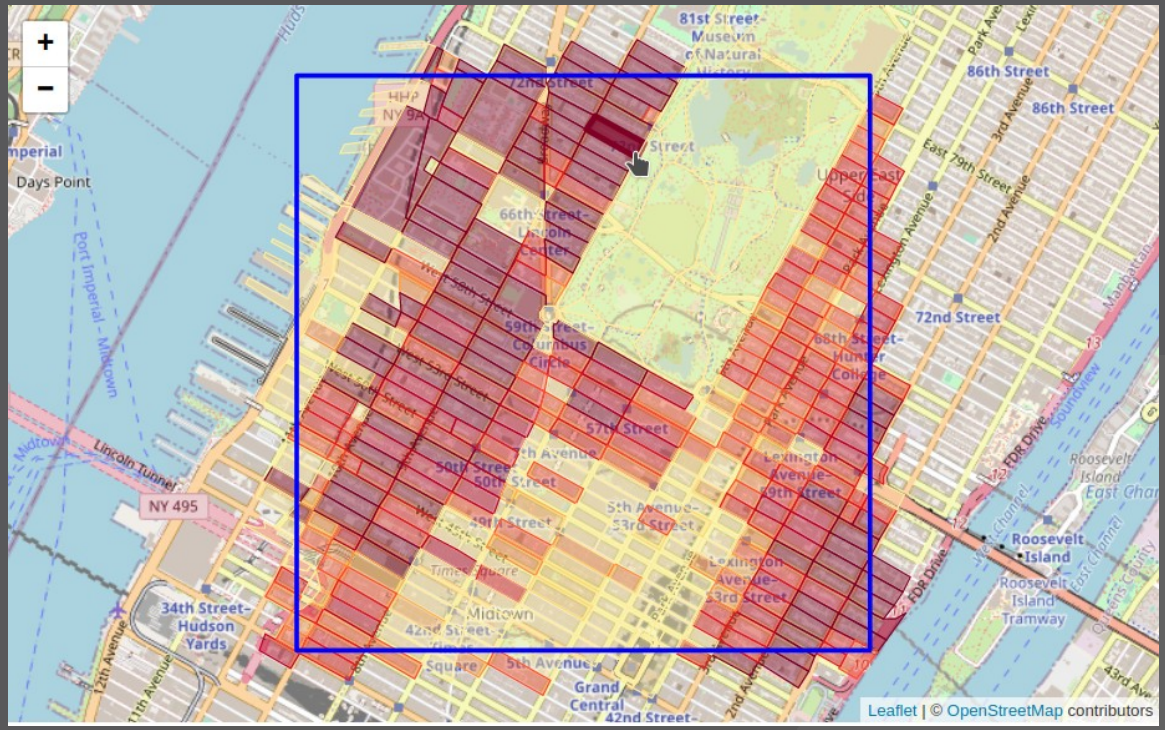
Creating und reading FlatGeobuf

- **GDAL 3.1:**

  ```
  ogr2ogr -f FlatGeobuf countries.fgb countries.json
  ```

- **Supported applications:**
  - OpenLayers, Leaflet, GeoServer (WFS output format), QGIS
- **Programming languanges:**
  - JavaScript, TypeScript, C++, C#, Java, Rust

FlatGeobuf demo



- 11GB census data
- https://flatgeobuf.org/examples/leaflet/large.html

SQLite over HTTP

- **absurd-sql**
  - Experimental SQLite access with HTTP Range requests
  - Backend for sql.js (sqlite3 compiled for the web)
  - Read and write access
  - https://github.com/jlongster/absurd-sql

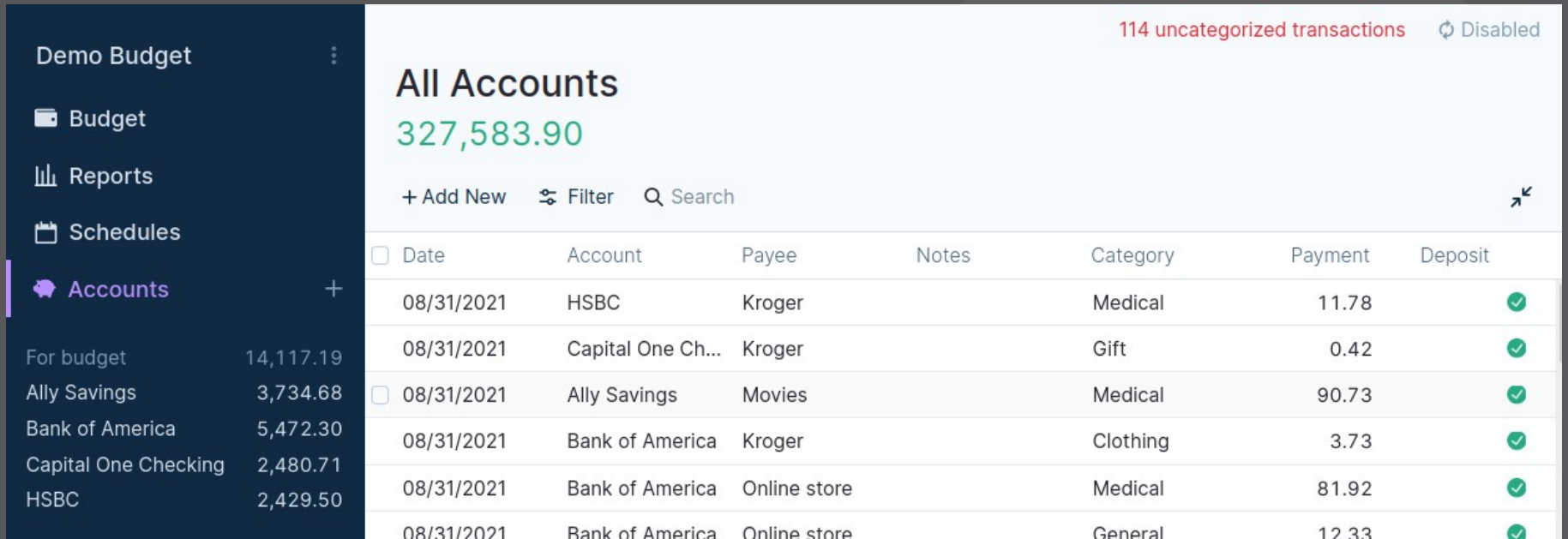
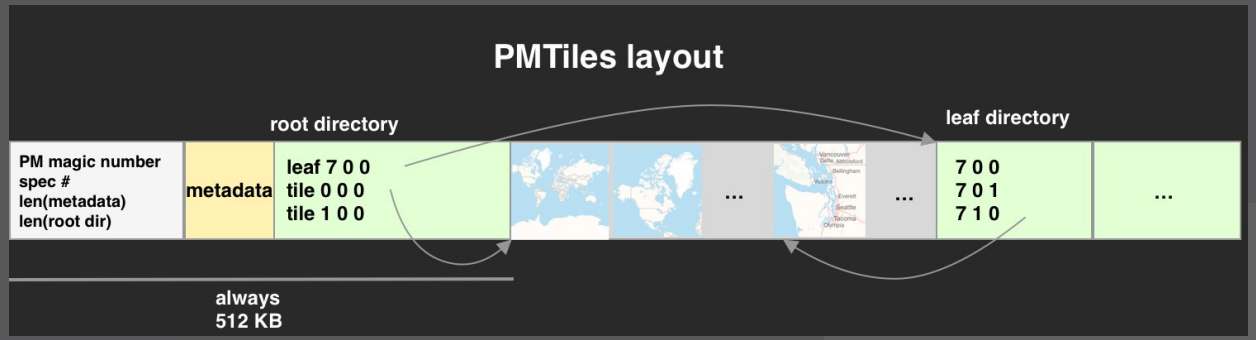


Exhibit #3: PMTiles

- Single-file archive format for pyramids of map tiles



- Advantages:
  - Raster and vector tiles
  - Optimized for HTTP Range requests
  - Fast file-based access like MBTiles
- https://github.com/protomaps/PMTiles

Exhibit #4: Point cloud data

Entwine Point Tile (EPT)

- Octree-based storage of LAZ files
- JSON metadata

COPC – Cloud Optimized Point Cloud:

- Include metadata in LAZ header
- Support for single file or external octree storage
- https://copc.io/ (Draft spec)

Summary

- Raster → COG
- Vector → FlatGeobuf
- Tiles → PMTiles
- Point clouds → COPC

Thank you!

Pirmin Kalberer
@implgeo