



FOSS4G 2021



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?

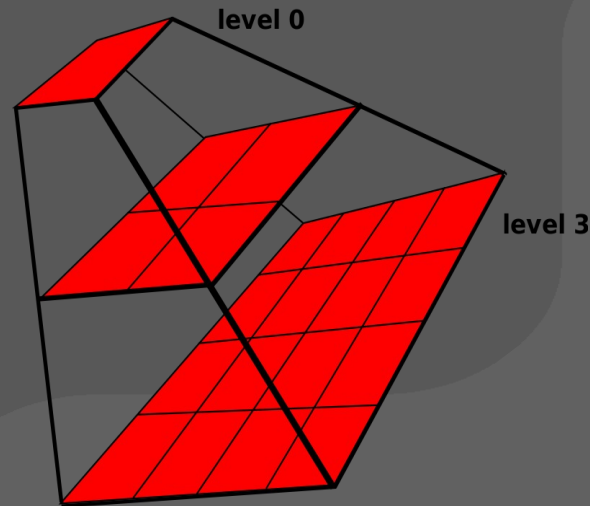
- Geospatial data files can be very big
- Reading files over a network much slower than from local disc
- Loading the whole file may be unfeasible





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
```



GeoTIFF:

- › Supports tiles
- › Supports overviews

Cloud Optimized GeoTIFF (COG):

- › Ordered Metadata
- › Ordered imagery
- › <https://www.cogeo.org/>





➤ **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
```





- › **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>

gt.js



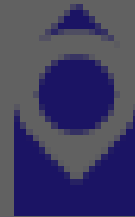
Exhibit #2: Vector data



- › 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/>





› GDAL 3.1:

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

› Supported applications:

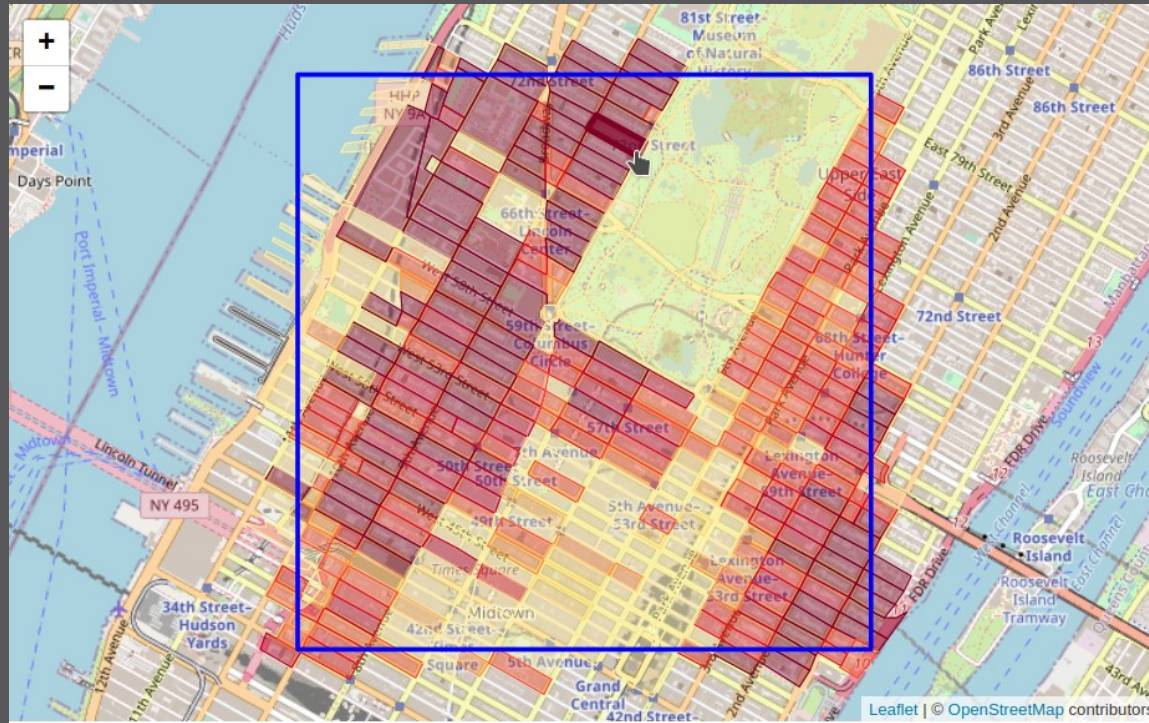
- › OpenLayers, Leaflet, GeoServer (WFS output format), QGIS

› Programming languages:

- › JavaScript, TypeScript, C++, C#, Java, Rust



FlatGeobuf demo



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



› absurd-sql

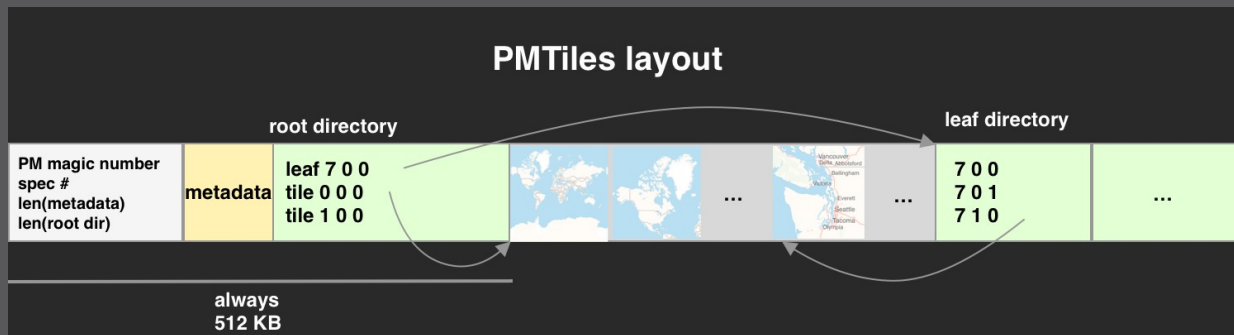
- › Experimental SQLite access with HTTP Range requests
- › Backend for sql.js (sqlite3 compiled for the web)
- › Read and write access
- › <https://github.com/jlongster/absurd-sql>

The screenshot shows a budgeting application interface. On the left is a dark sidebar with navigation options: Demo Budget, Budget, Reports, Schedules, and Accounts (highlighted). Below Accounts, a list of accounts and their balances is shown: For budget (14,117.19), Ally Savings (3,734.68), Bank of America (5,472.30), Capital One Checking (2,480.71), and HSBC (2,429.50). The main content area is titled 'All Accounts' and shows a total balance of 327,583.90. It includes controls for '+ Add New', 'Filter', and 'Search'. A table of transactions is displayed with columns for Date, Account, Payee, Notes, Category, Payment, and Deposit. The table shows several transactions from 08/31/2021, including payments to Kroger (Medical), Capital One (Gift), Ally Savings (Medical), Bank of America (Clothing), and Bank of America (Online store). A red notification at the top right indicates '114 uncategorized transactions' and a 'Disabled' status.

Date	Account	Payee	Notes	Category	Payment	Deposit
08/31/2021	HSBC	Kroger		Medical	11.78	✓
08/31/2021	Capital One Ch...	Kroger		Gift	0.42	✓
08/31/2021	Ally Savings	Movies		Medical	90.73	✓
08/31/2021	Bank of America	Kroger		Clothing	3.73	✓
08/31/2021	Bank of America	Online store		Medical	81.92	✓
08/31/2021	Bank of America	Online store		General	12.33	✓



- 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)



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



Thank you!



Pirmin Kalberer
@implgeo