State of GDAL
GDAL 3.8 & 3.9

Even Rouault
SPATIALYS

July 3rd 2024

GDAL/OGR : Introduction

- GDAL? Geospatial Data Abstraction Library. The swiss army knife for geospatial.
- Read and write Raster (GDAL) and Vector (OGR) datasets
- 250 (mainly) geospatial formats and protocols.
- Widely used

(> 100 http://trac.osgeo.org/gdal/wiki/SoftwareUsingGdal)

- MIT Open Source license (permissive)

OGC Features and Geometries JSON (JSON-FG)

- In-development spec extending GeoJSON:
  - use CRS other than WGS84 (“coordRefSys”)
  ⇒ “place” element in addition to “geometry”
  - support for solids and prisms as geometry types (probably curves in final version)
  - encoding of temporal characteristics of a feature (“time”)
  - ability to declare the type (“featureType”) and the schema of a feature (“featureSchema”).

Spec at https://github.com/opengeospatial/ogc-feat-geo-json

OGC Features and Geometries JSON (JSON-FG)


```
{
  "type": "FeatureCollection",
  "conformsTo" : [ "[ogc-json-fg-1-0.3:core]" ],
  "coordRefSys": "[EPSG:32631]",
  "features": [ {
        "type": "Feature",
        "id": 1,
        "featureType": "MyFeatureType",
        "featureSchema": "https://example.com/collections/MyFeatureType/schema",
        "geometry": { "type": "Point", "coordinates": [2, 49] },
        "place": { "type": "Point", "coordinates": [426857.988, 5427937.523] },
        "properties": { "foo": 1 },
        "time": {"timestamp": "2023-06-05T12:34:56Z"}
  } ]
}
```

OGC Features and Geometries JSON (JSON-FG)

- JSONFG driver shares similar behavior as GeoJSON one when applicable
- On writing, the driver handles filling both the “place” geometry with the native CRS, and automatic filling of “geometry” reprojected to WGS 84
- Multiple layers can be read and written, using the “featureType” special attribute
- Mapping between the “time” element and OGR feature properties
- Minimum read support for Polyhedron geometries (with a single outer shell) and Prism with Point, LineString or Polygon base
- Driver doc at https://gdal.org/drivers/vector/jsonfg.html

OGC Features and Geometries JSON (JSON-FG)

```
$ ogrinfo test.json -al
INFO: Open of `test.json' using driver `JSONFG' successful.

Layer name: MyFeatureType
Geometry: Point
Feature Count: 1
Extent: (426857.988000, 5427937.523000) - (426857.988000, 5427937.523000)
Layer SRS WKT: PROJCRS["WGS 84 / UTM zone 31N",[...],ID["EPSG",32631]]
Data axis to CRS axis mapping: 1,2
time: DateTime
foo: Integer (0.0)
OGRFeature(MyFeatureType):1
  time (DateTime) = 2023/06/05 12:34:56+00
  foo (Integer) = 1
  POINT (426857.988 5427937.523)
```

PMTiles (ProtoMap Tiles) v3

- Cloud-friendy tile container that enables to serve tiles efficiently with only object storage functionality
- Same spirit as COG or FlatGeoBuf
- Similar content as MBTiles, but with a highly optimized index / directory
- https://www.youtube.com/watch?v=zpQMLLDAowM : “Serverless Planet-scale Geospatial with Protomaps and PMTiles” - Brandon Liu - FOSS4G 2023 Prizren
- OGR driver has read/write support for vector tiles in MVT (Mapbox Vector Tiles) format
- Same options as existing MBTiles and MVT drivers
- /vsipmtiles/ virtual file system
- Doc: https://gdal.org/drivers/vector/pmtiles.html

Bathymetric related raster drivers: S-102, S-104, S-111

- IHO (International Hydrography Organization) standards
- Based on S-100 abstract specification
- HDF5 based containers
- Read-only drivers
- S-102: Bathymetric Surface Product (similar to existing BAG - Bathymetry Attributed Grid): depth and uncertainty
- S-104: Water Level Information for Surface Navigation Product: water level height and trend, multiple timestamps
- S-111: Surface Currents Product: current speed and direction, multiple timestamps

gdal_footprint command line utility

- Compute polygonal envelope of a raster
- Take into account nodata/alpha band
- ~= gdal_polygonize with specific options
- Decide how to combine validity of bands
- Can work on overviews for speed-up
- Several geometry processing options:
  - Reproject to another CRS
  - Densify or simplify (minimum distance of maximum number of points) polygons
  - Split multipolygons
  - Remove too small areas
- GDALFootprint() in C, gdal.Footprint() in Python

GDAL raster Tile Index (GTI) driver: virtual mosaics

- Improved version of VRT (Virtual RasTer)
- Handle very large collections of tiles (100K+)
- Any OGR vector driver can be a backend, but more efficient with GeoPackage, FlatGeoBuf, PostGIS
- Advantages over VRT:
  - Efficient on opening and pixel extraction even with very large collections
  - Smaller indices files
  - Use of spatial indices
  - On-the-fly reprojection
  - Z-order control (dedicated field)
  - Use of alpha band for overlapping sources

GDAL Raster Tile Index (GTI) driver: virtual mosaics

- Can be generated with gdaltindex, or programmatically
- A GTI tile index requires:
  - A vector layer with a column with the dataset location and its polygonal footprint
  - Global metadata describing:
    - Resolution
    - Extent
    - CRS
    - Data type
    - Number of bands
    - …

GDAL Raster Tile Index (GTI) driver: virtual mosaics

- Metadata can be embedded in formats allowing it (GeoPackage, FlatGeoBuf, PostGIS), or provided in a dedicated small XML file
- gdaltindex -gti_filename index.xml -lyr_name index
  -t_srs EPSG:26711 -tr 60 60 index.gti.fgb $PWD/*.tif

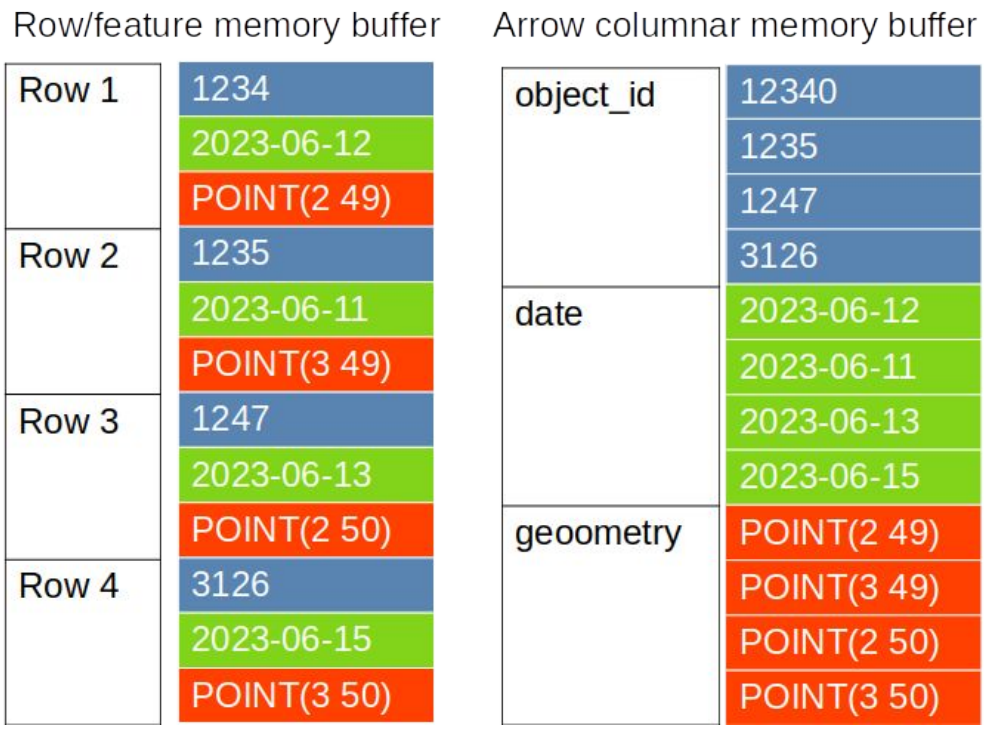
  ⇒ Index.xml:

```
<GDALTileIndexDataset>
  <IndexDataset>index.gti.fgb</IndexDataset>
  <IndexLayer>index</IndexLayer>
  <LocationField>location</LocationField>
  <ResX>60</ResX>
  <ResY>60</ResY>
</GDALTileIndexDataset>
```

- All details at https://gdal.org/drivers/raster/gti.html

Arrow interface: quick recap

- GDAL 3.6 introduced a Arrow-based columnar oriented read API for vector features



Enhancements in Arrow interface (GDAL 3.8)

- Parquet driver: enhancements in attribute and spatial filtering handling on the read side
- Arrow compatible interface available on the write side with a OGRLayer::WriteArrowBatch()
  - Generic implementation for all drivers
  - Specialized implementation in Arrow and Parquet drivers
  - Ogr2ogr uses in simple translation cases Arrow read & write capabilities for faster execution, when source dataset has an optimized Arrow read interface
    - GeoPackage -> Parquet: 3x faster
    - Parquet -> Parquet: 10x faster

Enhancements in (Geo)Parquet driver

- Support/reading nested list/map datatypes as JSON
- Implement full spatial filtering (not just bbox intersection
- GeoParquet 1.1 features (GDAL 3.9):
  - Bounding box columns per feature for fast spatial filtering (using Parquet statistics)
  - On creation, option to sort features spatially for more efficient grouping
  - Alternate GeoArrow encoding

Enhanced support for geometry coordinate precision (GDAL 3.9, RFC 99)

- Unified framework to specify geometry coordinate precision: https://gdal.org/development/rfc/rfc99_geometry_coordinate_precision.html
- Formats enhanced to store coordinate precision: GeoJSON, JSON-FG, GML, CSV, GeoPackage
- GeoPackage can perform optional binary coordinate precision, to combine with lossless compression (ZIP)
- ogrinfo (in JSON output) reports coordinate precision if known
- Ogr2ogr: specify precision or propagate source coordinate precision

GDAL driver plugin related enhancements

- Drivers that depend on external libraries (in particular proprietary SDKs) can be built as separate, run-time loadable libraries
- Used for example by the Alpine Linux official GDAL package or conda-forge GDAL build for Parquet driver
- Enhancement in GDAL 3.9 to only load those plugin drivers when strictly needed
- Speed enhancement, especially for short lived process
- Details at https://gdal.org/development/rfc/rfc96_deferred_plugin_loading.html

Miscellaneous

- New driver for vector Miramon format

- TileDB: read/write support for multidimensional API
- Performance improvements in GeoPackage: spatial index creation ⇒ 3 to 4 times faster
- Line of sight algorithm (C / Python API)
- Update of build requirements for GDAL >= 3.9 to C++17 and third-party libraries as available in Ubuntu >= 20.04 (GDAL C++ API still only requiring C++11)

  Cf https://gdal.org/development/rfc/rfc98_build_requirements_gdal_3_9.html
- Use of a C++ command line argument parsing framework (argparse) (in-progress GDAL 3.9 / GDAL 3.10)

Miscellaneous

- gdaladdo enhancements to partially refresh existing overviews:
  - --partial-refresh-from-source-timestamp
  - --partial-refresh-from-projwin <ulx> <uly> <lrx> <lry>
  - --partial-refresh-from-source-extent <filename1,...,filenameN>
- Multiple enhancements to vrt:// connection string, covering most options of gdal_translate

  E.g. “vrt://my.tif?srcwin=2,50,3,49”
- Various improvements in Python bindings to reduce long-standing “gotchas” related to cross-object references.

GDAL 3.10 preview

- GeoParquet: attribute and spatial filter push down for multi-file datasets
- TileDB: support for nodata and overviews
- Performance improvements in gdal_viewshed (multi-threading)
- Partial support for 64-bit ObjectIDs in OpenFileGDB driver
- XODR: new vector driver to read road networks in OpenDrive format
- Probable support for Float16/Half-Float data type in Zarr format

Thanks to GDAL sponsors! (gdal.org/sponsors)

- Gold level:

- Silver level:

- Bronze level:

- Supporter level:

Vortex f.d.c. Regrid Satelligence Kaplan Open Source Consulting Umbra Dynamic Graphics, Inc. PIX4D Space Intelligence Myles Sutherland

Questions?

Links:

http://gdal.org/

Contact: even.rouault@spatialys.com