State of GDAL
GDAL 3.6 & 3.7

Even Rouault
SPATIALYS

June 28th 2023
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


- MIT Open Source license (permissive)
CMake build system

- In GDAL 3.5, CMake as a new build system to replace long time existing autotools-based and Visual Studio’s nmake ones
- Since GDAL 3.6, CMake is now only the build system supported
- Many integrations available with build systems such as vcpkg or conda-forge
- Extensive documentation of all the options at https://gdal.org/development/building_from_source.html
(Open)FileGeodatabase enhancements

- Full write and update support for FileGeodatabase vector, in core GDAL 3.6:
  - Creation of new datasets
  - Addition/deletion of vector layers
  - Addition/update/deletion of attributes
  - Addition/update/deletion of features
  - Creation and update of spatial index
  - Creation and update of attribute index
  - Creation/management of field domains
  - Creation and management of layer relationships
  - Database compaction
OpenFileGDB integration in QGIS 3.28
(Open)FileGeodatabase enhancements

- Prior O.S. effort: ArcRaster Rescue (R. Barnes)
- Read support for FileGeodatabase v10 raster datasets, in core GDAL 3.7
  - Read CRS and georeferencing information
  - Tile-based reading
  - Expose overviews/pyramids
  - Compression methods: uncompressed, LZ77 (Deflate), JPEG and JPEG2000
  - Value attribute tables
- Read support for FileGDB v9 for GDAL 3.8
Arrow-based columnar oriented read API

Columnar formats:
- Parquet
- Arrow
- TileDB

Tools:
- pandas
- GeoPandas
- Apache Parquet
- R-spatial / sf-arrow
- [tile]DB
- DuckDB
- GDAL
- Spark
Column-oriented = information for a given attribute is grouped by many rows

<table>
<thead>
<tr>
<th>object_id</th>
<th>date</th>
<th>geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>2023-06-12</td>
<td>POINT(2 49)</td>
</tr>
<tr>
<td>1235</td>
<td>2023-06-11</td>
<td>POINT(3 49)</td>
</tr>
<tr>
<td>1247</td>
<td>2023-06-13</td>
<td>POINT(2 50)</td>
</tr>
<tr>
<td>3126</td>
<td>2023-06-15</td>
<td>POINT(3 50)</td>
</tr>
</tbody>
</table>

Row/feature memory buffer:

<table>
<thead>
<tr>
<th>object_id</th>
<th>date</th>
<th>geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>2023-06-12</td>
<td>POINT(2 49)</td>
</tr>
<tr>
<td>1235</td>
<td>2023-06-11</td>
<td>POINT(3 49)</td>
</tr>
<tr>
<td>1247</td>
<td>2023-06-13</td>
<td>POINT(2 50)</td>
</tr>
<tr>
<td>3126</td>
<td>2023-06-15</td>
<td>POINT(3 50)</td>
</tr>
</tbody>
</table>

Arrow columnar memory buffer:

<table>
<thead>
<tr>
<th>object_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>12340</td>
</tr>
<tr>
<td>1235</td>
</tr>
<tr>
<td>1247</td>
</tr>
<tr>
<td>3126</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2023-06-12</td>
</tr>
<tr>
<td>2023-06-11</td>
</tr>
<tr>
<td>2023-06-13</td>
</tr>
<tr>
<td>2023-06-15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINT(2 49)</td>
</tr>
<tr>
<td>POINT(3 49)</td>
</tr>
<tr>
<td>POINT(2 50)</td>
</tr>
<tr>
<td>POINT(3 50)</td>
</tr>
</tbody>
</table>
Arrow-based columnar oriented read API

- [https://gdal.org/development/rfc/rfc86_column_oriented_api.html](https://gdal.org/development/rfc/rfc86_column_oriented_api.html)
- OGRLayer::GetArrowStream() → ArrowStream
- Arrow Stream C interface:
  - get_schema(stream) → ArrowSchema
  - get_next(stream) → ArrowArray
- ds = ogr.Open("my.gpkg")
  lyr = ds.GetLayer(0)
  for batch in lyr.GetArrowStreamAsPyArrow():
      … do something with batch …
  for batch in lyr.GetArrowStreamAsNumPy():
      … do something with batch …
Arrow-based columnar oriented read API

● Generic implementation for all other OGR drivers
  ○ uses feature-based iteration underneath (GetNextFeature())
  ○ of course adds some overhead
● Efficient implementation in drivers:
  ○ (Geo)Parquet/(GeoArrow)
  ○ TileDB (vector side added in GDAL 3.7)
  ○ GeoPackage
  ○ FlatGeoBuf
Arrow-based columnar oriented read API

- Benchmarks to load a 3.2 million rows (footprint polygons) of 13 attributes each

<table>
<thead>
<tr>
<th></th>
<th>GeoParquet</th>
<th>TileDB</th>
<th>GeoPackage</th>
<th>FlatGeoBuf</th>
<th>Shapefile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File size</strong></td>
<td>0.43 GB</td>
<td>1.1 GB</td>
<td>1.67 GB</td>
<td>1.8 GB</td>
<td>2.92 GB</td>
</tr>
<tr>
<td><strong>Feature-based iteration</strong></td>
<td>6.2 s</td>
<td>6.0 s</td>
<td>6.6 s</td>
<td>5.0 s</td>
<td>10.3 s</td>
</tr>
<tr>
<td><strong>Arrow/Columnar (1 thread)</strong></td>
<td>1.6 s (x 3.9 speedup)</td>
<td>---</td>
<td>2.2 s (x 3 speedup)</td>
<td>3.0 s (x 1.7)</td>
<td>16.8 s (generic impl.) (x 0.61)</td>
</tr>
<tr>
<td><strong>Arrow/Columnar (4 threads)</strong></td>
<td>1.0 s (x 6.2 speedup)</td>
<td>1.3 s (x 4.6)</td>
<td>0.7 s (x 9.4)</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>
New drivers (GDAL 3.6)

- BASISU and KTX2 read/write raster drivers:
  - Basis Universal Textures: GPU-optimized
  - Khronos Texture Container 2.0
  - Require (forked) basisu open source library

- JPEG-XL standalone read/write driver:
  - JPEG-XL: competitor to AVIF/HEIF/WEBP
  - Lossless and lossy profiles
  - Already available as a GeoTIFF codec
  - Requires open source libjxl library
New drivers (GDAL 3.7)

- **NSIDCbin raster read-only driver:**
  - NSIDC=National Snow and Ice Data Centre
  - Sea Ice Concentrations
  - Raw binary format
  - Daily and monthly maps in the north and south hemispheres

- **GTFS vector read-only driver:**
  - General Transit Feed Specification
  - Common format for public transportation schedules and associated geographic information
  - Expose text tables augmented with geometries
ogrinfo : API

- Available through the GDALVectorInfo() C API call
- gdal.VectorInfo( dataset / filename ) in Python
ogrinfo: … and JSON output!

$ ogrinfo -json poly.shp
{
  "layers": [
    {
      "name": "poly",
      "metadata": {
      }
    },
    "geometryFields": [
      {
        "name": "", 
        "type": "Polygon",
        "extent": [
          478315.53125, 
          4762880.5, 
          481645.3125, 
          4765610.5
        ],
        "coordinateSystem": {
          [...]
        }
      },
      "featureCount": 10,
      "fields": [
        {
          "name": "AREA",
          "type": "Real",
          "width": 12,
          "precision": 3,
          "nullable": true,
          "uniqueConstraint": false
        },
        {
          "name": "EAS_ID",
          "type": "Integer64",
          "width": 11,
          "nullable": true,
          "uniqueConstraint": false
        },
        {
          "name": "PRFEDEA",
          "type": "String",
          "width": 16,
          "nullable": true,
          "uniqueConstraint": false
        }
      ]
    }
  ]
}
Miscellaneous

- `ogr_layer_algebra.py`: promoted as a supported utility
- `OGRLayer::UpsertFeature()` and `UpdateFeature()`
- PNG decompression: 1.7 to 2.0 times faster when building GDAL against libdeflate
- Direct access to compressed raster data: extraction of JPEG/JPEGXL/WEBP/etc. Tiles:
  - Implemented in GeoTIFF, VRT, JPEG, JPEGXL and WEBP drivers
    - [https://gdal.org/development/rfc/rfc90_read_compressed_data.html](https://gdal.org/development/rfc/rfc90_read_compressed_data.html)
# GeoTIFF multi-threaded decompression

- Enabled if GDAL_NUM_THREADS configuration option/env. variable is set
- 3-band Byte tiled 10,000 x 5,000 large file:

<table>
<thead>
<tr>
<th>Compression</th>
<th>Time (ms) single-threaded</th>
<th>Time (ms) 12 threads</th>
<th>Speed-up factor</th>
<th>File size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>210</td>
<td>112</td>
<td>1.9</td>
<td>157</td>
</tr>
<tr>
<td>Deflate</td>
<td>310</td>
<td>137</td>
<td>2.3</td>
<td>11</td>
</tr>
<tr>
<td>JPEG (lossy)</td>
<td>312</td>
<td>138</td>
<td>2.3</td>
<td>1.1</td>
</tr>
<tr>
<td>LZW</td>
<td>390</td>
<td>142</td>
<td>2.7</td>
<td>12</td>
</tr>
<tr>
<td>WebP lossy</td>
<td>411</td>
<td>148</td>
<td>2.8</td>
<td>0.5</td>
</tr>
<tr>
<td>WebP lossless</td>
<td>558</td>
<td>157</td>
<td>3.6</td>
<td>7.7</td>
</tr>
<tr>
<td>JPEG-XL lossy</td>
<td>1415</td>
<td>368</td>
<td>3.8</td>
<td>0.7</td>
</tr>
<tr>
<td>JPEG-XL lossless</td>
<td>6810</td>
<td>1368</td>
<td>5.0</td>
<td>8.6</td>
</tr>
</tbody>
</table>

- 11977 x 8745 x 8 bands UInt16 tiled deflate compressed file

<table>
<thead>
<tr>
<th>Compression</th>
<th>Time (ms) single-threaded</th>
<th>Time (ms) 12 threads</th>
<th>Speed-up factor</th>
<th>File size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deflate</td>
<td>5700</td>
<td>1109</td>
<td>5.1</td>
<td>990</td>
</tr>
</tbody>
</table>
SOZIP: Seek-optimized ZIP format

- Open-specification on top of venerable .zip format
- Random access in large compressed files directly from a .zip file without prior decompression
- Fully backwards compatible
- Use cases: GeoPackage, FlatGeoBuf, Shapefile, …
- Available in GDAL 3.7 / “sozip” command line utility

⇒ More at 4h30 today at “SOZip: using directly (geospatial) large compressed files in a ZIP archive!” session
GDAL 3.8 preview

- New vector driver to read and write the OGC Feature and Geometries JSON (JSON-FG) candidate standard
- New vector driver to read and write PMTiles dataset with vector tile. PMTiles: cloud-friendly tile container
- Multi-dimensional API in TileDB driver
Thanks to GDAL sponsors!

- Gold level:
  - Maxar
  - AWS
  - Planet
  - Esri
  - Microsoft

- Silver level:
  - Google
  - Safe Software

- Bronze level:
  - L3Harris
  - Geo
  - MapGears
  - GeoCzech
  - Koordinates
  - Aerometrex
  - Frontier SI

- Supporter level:
  - Umbra
  - Space Intelligence
  - PIX4D
  - Kaplan Open Source Consulting
  - Myles Sutherland
  - Regrid
Questions?

Links:

http://gdal.org/

Contact: even.rouault@spatialys.com