tilezilla Documentation

Release 0.1.0

Chris Holden

January 19, 2017

Contents

Todo

Brief introduction

User Guide:

CHAPTER 1

Configuration

1.1 Tile Specification

A tile specification describes the positioning, grid layout, and coordinate reference system that ingested imagery will be reprojected, diced, and aligned to match with.

A tile specification is described by four attributes:

- `crs` (`str`): The coordinate reference system, given as a Proj4 string or Well-Known-Text.
- `ul` (`tuple`): The upper left X and Y coordinates of the tile specification
- `res` (`tuple`): The pixel resolution of data in the tile specification
- `size` (`tuple`): The number of pixels (columns and rows) in each tile. The total area covered by a tile is a function of the pixel size and the number of pixels per tile.

Tilezilla includes a few "well known specifications" that include:

- **WELD_CONUS**
  - `crs`: "EPSG:5070"
  - `ul`: [-2565600.0, 3314800.0]
  - `res`: [30, 30]
  - `size`: [5000, 5000]

1.2 Database

Tilezilla's database backend is written using SQLAlchemy ORM to abstract as much as possible away from the type of database used. Currently, only SQLite has been tested but the structure should be in place to switch to something more complex (e.g., PostgreSQL).

1.2.1 SQLite

Todo

Describe SQLite configuration

1.3 Storage

Tiled data are stored on disk either as a collection of GeoTIFF images or NetCDF4 files.

Currently, only the GeoTIFF storage format is implemented.

1.3.1 GeoTIFF

Todo

Document the GeoTIFF storage format

1.4 Products

Todo

Write about existing products (ESPALandsat) and how one can write YAML file to describe arbitrary products for ingest

1.5 Example

The following example configuration file describes the configuration needed to ingest Landsat data processed to surface reflectance through the ESPA system into a tiling scheme that has the same parameters as the Web Enabled Landsat Data (WELD) tiled data product.

The `include_filter` option in the `products` section limits the ingest to only include surface reflectance, brightness temperature, and CFmask bands. Any reprojection necessary for this ingest will be done using nearest neighbor resampling.

```
# Metadata
version: 0.1.0

# Database
# See connection information at:
# http://docs.sqlalchemy.org/en/latest/core/engines.html#sqlalchemy.engine.url.URL
database:
    drivername: sqlite
    database: /home/ceholden/tiles/tilezilla.db
    # Not required, but used with postgresql, etc.
    # username:
    # password:
    # host:
    # port:
    debug: False

# Tile storage method
store:
    ## Required
    name: GeoTIFF  # could be NetCDF
```

```
    root: /home/ceholden/tiles
    tile_dirpattern: 'h{horizontal:04d}v{vertical:04d}'
    tile_imgpattern: '{product.timeseries_id}_{band.standard_name}.tif'
    ## Additional option -- specific (?) to GeoTIFF: creation options
    co:
        tiled: true
        blockxsize: 256
        blockysize: 256
        compress: deflate


# Tile specification
## For recognized systems
tilespec: WELD_CONUS
## Or manually
# tilespec:
#     # Coordinate reference system
#     crs: EPSG:5070
#     # Upper left x/y coordinate
#     ul: 15, 15
#     # Resolution (x/y) of each pixel
#     res: 30, 30
#     # Number of pixels (x/y) in each tile
#     size: 500, 500


# Products
products:
    ## Product handling options, specified by product type
    ESPALandsat:
        include_filter:
            regex: false
            # Attributes of each product band to include
            long_name:
                - '*surface reflectance*'
                - '*brightness temperature*'
                - '*cfmask_band*'
        resampling: nearest
```

CHAPTER 2

Command Line Interface (CLI)

Note: All of the usage examples assume the Tilezilla configuration file has been specified by defining an environment variable, `TILEZILLA_CONFIG`. For example, using the Bash shell:

```
export TILEZILLA_CONFIG=/path/to/configuration_file.yaml
```

Specifying using an environment variable takes the place of `'tilez -C <config_file>` or `tilez --config <config_file>`.

2.1 Ingest

The `tilez ingest` program slices dataset product images into tiles, writes these tiled images to disk, and then indexes metadata about these data into a database for future use.

2.1.1 Usage

```
> tilez ingest --help
Usage: tilez ingest [OPTIONS] [SOURCES]...

Options:
  -pe, --parallel-executor [serial|process]
                          Method of parallel execution
  -j, --njob INTEGER      Number of jobs for parallel execution
  --log_dir PATH          Log ingests to this directory (otherwise to
                          stdout)
  --overwrite                   Overwriting existing tiled data
  -h, --help                    Show this message and exit.
```

Note: All of the usage examples assume the Tilezilla configuration file has been specified by defining an environment variable, `TILEZILLA_CONFIG`. For example, using the Bash shell:

```
export TILEZILLA_CONFIG=/path/to/configuration_file.yaml
```

Specifying using an environment variable takes the place of `'tilez -C <config_file>` or `tilez --config <config_file>`.

2.1.2 Examples

Todo

Demo of `tilez ingest`

2.2 Database Queries

Datasets ingested through Tilezilla are indexed in a database (see *Database*). The `tilez db` command and subcommands allows the user to easily query this database without requiring knowledge of SQL.

2.2.1 Examples

Note: All of the usage examples assume the Tilezilla configuration file has been specified by defining an environment variable, `TILEZILLA_CONFIG`. For example, using the Bash shell:

```
export TILEZILLA_CONFIG=/path/to/configuration_file.yaml
```

Specifying using an environment variable takes the place of `'tilez -C <config_file>` or `tilez --config <config_file>`.

1. Find information about the *products* table

```
> tilez db info product
11:30:36:INFO:*   Information about: <class 'tilezilla.db.sqlite.tables.TableProduct'>
11:30:36:INFO:==> Number of entries: 6
11:30:36:INFO:==> Enumerating columns in table: <class 'tilezilla.db.sqlite.tables.TableProduct'>
11:30:36:INFO:-       COLUMN #   NAME                TYPE
11:30:36:INFO:-       Col 00     "created"           DATETIME
11:30:36:INFO:-       Col 01     "updated"           DATETIME
11:30:36:INFO:-       Col 02     "id"                INTEGER (PRIMARY KEY)
11:30:36:INFO:-       Col 03     "ref_tile_id"       INTEGER
11:30:36:INFO:-       Col 04     "timeseries_id"     VARCHAR
11:30:36:INFO:-       Col 05     "platform"          VARCHAR
11:30:36:INFO:-       Col 06     "instrument"        VARCHAR
11:30:36:INFO:-       Col 07     "acquired"          DATETIME
11:30:36:INFO:-       Col 08     "processed"         DATETIME
11:30:36:INFO:-       Col 09     "metadata_"         TEXT
11:30:36:INFO:-       Col 10     "metadata_files_"   TEXT
11:30:36:INFO:-       HYBRID 00  "n_bands"           ?
```

2. Find products that weren't fully ingested

Sometimes a product ingest can fail. In order to help recover, we can query the database to find all bands from products that were partially ingested. In this example, we know that all products should contain 8 bands.

```
> tilez db search --filter "n_bands != 8" --group_by timeseries_id product
Searching table "product" where:
n_bands != 8
Results:
<ESPALandsat(timeseries_id=LT50130311995272AAA01, platform/instrument=LANDSAT_5/TM, acquired=1995-09-
<ESPALandsat(timeseries_id=LT50120311995345XXX01, platform/instrument=LANDSAT_5/TM, acquired=1995-12-
<ESPALandsat(timeseries_id=LT50120311996236XXX01, platform/instrument=LANDSAT_5/TM, acquired=1996-08-
<ESPALandsat(timeseries_id=LT50120311999116XXX01, platform/instrument=LANDSAT_5/TM, acquired=1999-04-
<ESPALandsat(timeseries_id=LT50120312011133EDC00, platform/instrument=LANDSAT_5/TM, acquired=2011-05-
<ESPALandsat(timeseries_id=LE70130312012119EDC00, platform/instrument=LANDSAT_7/ETM, acquired=2012-04
```

2.3 VRT Export

Tilezilla stores tiled data in a flexible and extendible manner so that additional data, including additional bands for the same acquisition, may be indexed and added. This storage arrangement, however, is not always ideal for analysis in existing codebases.

The `tilez spew` program was written to convert the storage systems used in Tilezilla (e.g., GeoTIFF) into something more familiar. Specifically, *tilez spew* will create multi-band Virtual Raster dataset (VRT) for each acquisition containing a collection of bands desired by the user. VRTs should ease the use of time series of imagery ingested by Tilezilla because the VRTs bundle multiple bands together.

```
$ tilez spew -h
Usage: tilez spew [OPTIONS] DESTINATION [PRODUCT_IDS]...

  Export tiled products to mutli-band VRTs for a given product ID

  Product IDs can be passed either as input arguments or through `stdin`
  using a pipe.

  By default the configuration file will be used to determine what bands are
  exported into the multi-band VRT format. If `--bands` is specified, the
  configuration file will be ignored and only bands specified by `--bands`
  will be exported.

Options:
  --bands TEXT  Override config file for bands to export into VRT (specify
                using band_attr:pattern)
  --regex       Allow patterns in `--bands` to be regular expressions
  -h, --help    Show this message and exit.
```

2.3.1 Example

Note: All of the usage examples assume the Tilezilla configuration file has been specified by defining an environment variable, `TILEZILLA_CONFIG`. For example, using the Bash shell:

```
export TILEZILLA_CONFIG=/path/to/configuration_file.yaml
```

Specifying using an environment variable takes the place of 'tilez -C <config_file> or tilez --config <config_file>.

To begin, the `spew` command needs to know what products in the database to convert to the VRT format.

```
> tilez db id --filter "n_bands = 8" --filter "acquired > 2000-01-01" --filter "acquired < 2016-01-01
9
```

Note that we use the command `tilez db id` as a proxy for `tilez db info --quiet --select id` and can include search filter options using the `--filter` flag. For more information on searching the database for ingested products, see *Database Queries*.

The fastest way of exporting the products found during a search into VRT format is to use the Unix pipe:

```
> tilez db id --filter "n_bands = 8" --filter "acquired > 2000-01-01" --filter "acquired < 2016-01-01
11:18:58:INFO:==> Beginning export to VRT for 9 products
11:18:58:INFO:-       Exporting product: LT50120312002300LGS01
11:18:58:INFO:-           Friendly band names: sr_band1, sr_band2, sr_band3, sr_band4, sr_band5, sr_b
...
11:18:58:INFO:-       Exporting product: LT50120312002300LGS01
11:18:58:INFO:-           Friendly band names: sr_band1, sr_band2, sr_band3, sr_band4, sr_band5, sr_b
```

Once complete, the command has created 9 VRTs that reference 72 product bands (9 tiled products x 8 bands/product) and organized them into the same hierarchical structure used to store the tiled data (e.g., by product, tile reference, and product ID):

```
> tree VRTs/
> VRTs
-- ESPALandsat
    -- h0029v0005
    |   -- LT50120312002300LGS01
    |       -- LT50120312002300LGS01.vrt
    -- h0029v0006
    |   -- LT50120312002300LGS01
    |       -- LT50120312002300LGS01.vrt
    -- h0029v0007
    |   -- LT50120312002300LGS01
    |       -- LT50120312002300LGS01.vrt
    -- h0030v0005
    |   -- LT50120312002300LGS01
    |       -- LT50120312002300LGS01.vrt
    -- h0030v0006
    |   -- LT50120312002300LGS01
    |       -- LT50120312002300LGS01.vrt
    -- h0030v0007
    |   -- LT50120312002300LGS01
    |       -- LT50120312002300LGS01.vrt
    -- h0031v0005
    |   -- LT50120312002300LGS01
    |       -- LT50120312002300LGS01.vrt
    -- h0031v0006
    |   -- LT50120312002300LGS01
    |       -- LT50120312002300LGS01.vrt
    -- h0031v0007
        -- LT50120312002300LGS01
            -- LT50120312002300LGS01.vrt
```

Each of these VRTs contains the 8 bands specified by the configuration file `include_filter` option for the `ESPALandsat` product. As such, a `gdalinfo` reveals the structure as:

```
> gdalinfo VRTs/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01.vrt
Driver: VRT/Virtual Raster
Files: VRTs/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01.vrt
       /home/ceholden/tiles/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01_sr_ban
```

```
       /home/ceholden/tiles/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01_sr_ban
       /home/ceholden/tiles/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01_sr_ban
       /home/ceholden/tiles/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01_sr_ban
       /home/ceholden/tiles/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01_sr_ban
       /home/ceholden/tiles/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01_sr_ban
       /home/ceholden/tiles/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01_toa_ba
       /home/ceholden/tiles/ESPALandsat/h0029v0005/LT50120312002300LGS01/LT50120312002300LGS01_cfmask
Size is 5000, 5000
Coordinate System is:
PROJCS["NAD83 / Conus Albers",
    GEOGCS["NAD83",
        DATUM["North_American_Datum_1983",
            SPHEROID["GRS 1980",6378137,298.257222101,
                AUTHORITY["EPSG","7019"]],
            TOWGS84[0,0,0,0,0,0,0],
            AUTHORITY["EPSG","6269"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4269"]],
    PROJECTION["Albers_Conic_Equal_Area"],
    PARAMETER["standard_parallel_1",29.5],
    PARAMETER["standard_parallel_2",45.5],
    PARAMETER["latitude_of_center",23],
    PARAMETER["longitude_of_center",-96],
    PARAMETER["false_easting",0],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AXIS["X",EAST],
    AXIS["Y",NORTH],
    AUTHORITY["EPSG","5070"]]
Origin = (1784400.000000000000000,2564800.000000000000000)
Pixel Size = (30.000000000000000,-30.000000000000000)
Corner Coordinates:
Upper Left  ( 1784400.000, 2564800.000) ( 73d24'28.78"W, 44d 9'21.00"N)
Lower Left  ( 1784400.000, 2414800.000) ( 73d50'33.58"W, 42d50'56.42"N)
Upper Right ( 1934400.000, 2564800.000) ( 71d35'17.96"W, 43d49'35.62"N)
Lower Right ( 1934400.000, 2414800.000) ( 72d 3'18.28"W, 42d31'35.86"N)
Center      ( 1859400.000, 2489800.000) ( 72d43'24.65"W, 43d20'32.69"N)
Band 1 Block=128x128 Type=Int16, ColorInterp=Blue
  NoData Value=-9999
Band 2 Block=128x128 Type=Int16, ColorInterp=Green
  NoData Value=-9999
Band 3 Block=128x128 Type=Int16, ColorInterp=Red
  NoData Value=-9999
Band 4 Block=128x128 Type=Int16, ColorInterp=Undefined
  NoData Value=-9999
Band 5 Block=128x128 Type=Int16, ColorInterp=Undefined
  NoData Value=-9999
Band 6 Block=128x128 Type=Int16, ColorInterp=Undefined
  NoData Value=-9999
Band 7 Block=128x128 Type=Int16, ColorInterp=Undefined
  NoData Value=-9999
Band 8 Block=128x128 Type=Byte, ColorInterp=Undefined
  NoData Value=255
```

One could also manually specify the bands to include in the VRT images manually using the *–bands* option:

```
> tilez -v spew --bands long_name="*surface reflectance*" SR_VRTS 1
11:20:20:INFO:==> Beginning export to VRT for 1 products
11:20:20:INFO:-       Exporting product: LT50120312002300LGS01
11:20:20:INFO:-           Friendly band names: sr_band1, sr_band2, sr_band3, sr_band4, sr_band5, sr_b
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search