Next Gen File Formats - Part II

a.k.a. Getting your hands-dirty with open-formats
a.k.a. Formats that make people happy

Materials for the workshop are available at
https://downloads.openmicroscopy.org/presentations/2020/Dundee/Workshops/NGFF

Software versions used for this workshop:

- conda: 4.8.3
- bioformats2raw: 0.2.2
- raw2ometiff: 0.2.4
- ome-zarr: 0.0.6
- napari: 0.3.1

Summary

⇒ Installation instructions (Best if done **before** the workshop)

⇒ Review overall workflow

⇒ Glencoe's pipeline for whole-slide imaging including `bioformats2raw`
- Standard conversion examples
- Zarr conversion examples

⇒ Investigating the raw intermediate format for public data
- Description of the format

⇒ Towards an ome-zarr library with an example napari plugin

Programme

Installation instructions

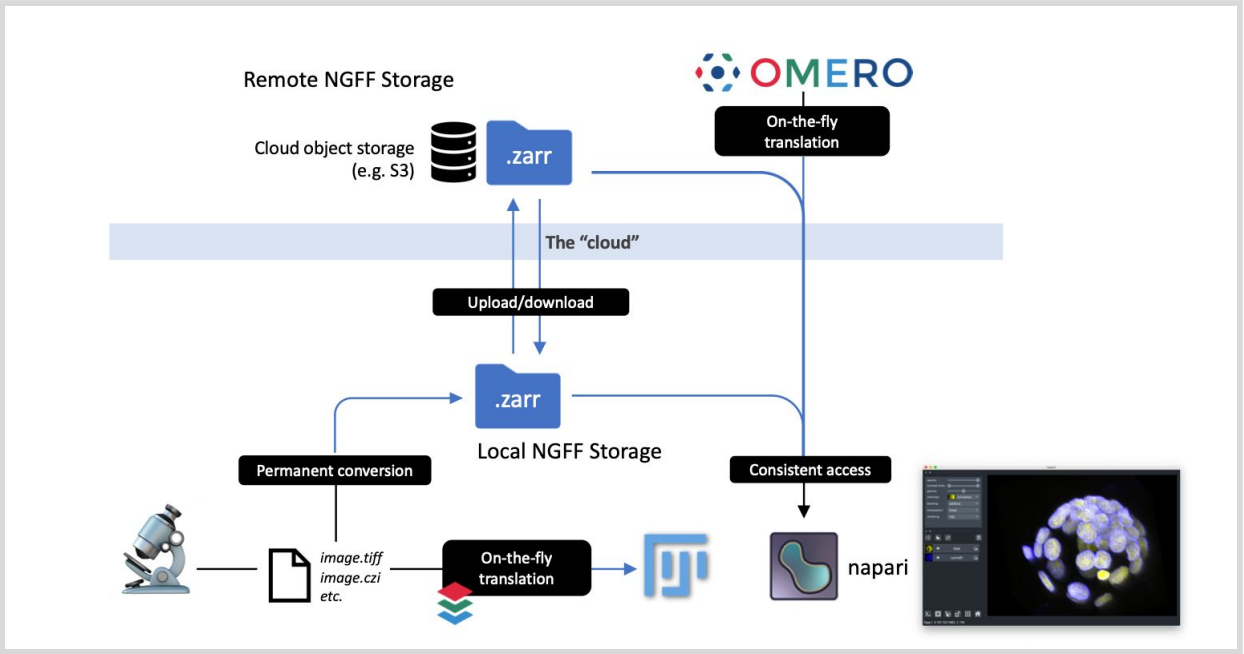
Easiest way to install the above tools is by creating a conda environment using the yaml file from [ome-zarr-guide](#) :

```
channels:
  - ome
  - conda-forge
  - defaults
dependencies:
  - python=3.6
  - bftools=6.3.1
  - bioformats2raw=0.2.2
  - raw2ometiff=0.2.4
  - blosc
  - napari
  - zarr
  - dask
  - s3fs
  - pip
  - pip:
    - ome-zarr
```

1. Follow the [conda documentation](#) if you have not previously intalled conda.
2. Place the above in a file named “environment.yml” and run:
   `conda env create -n demo -f environment.yml`

Alternatively, you can manually download [bioformats2raw](#) and [raw2ometiff](#), and install the necessary Python libraries into your (virtual-)environment of choice, e.g. with `pip install ome-zarr[napari]`

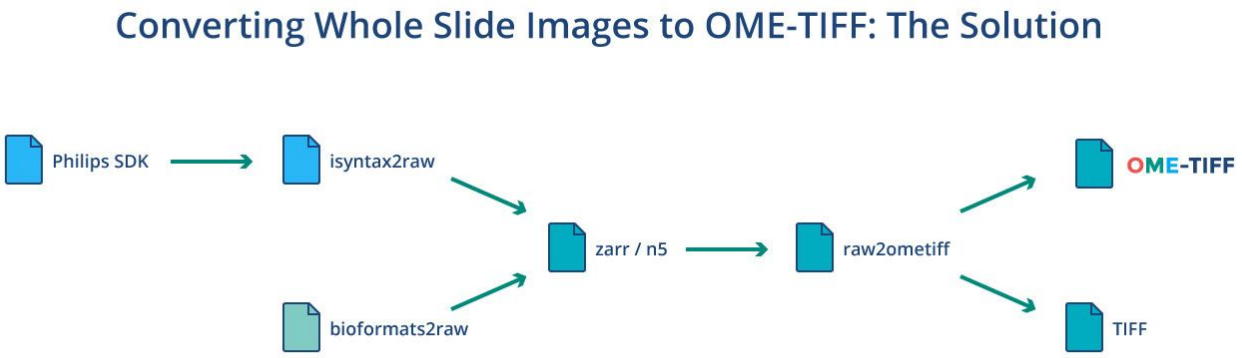
Workflows

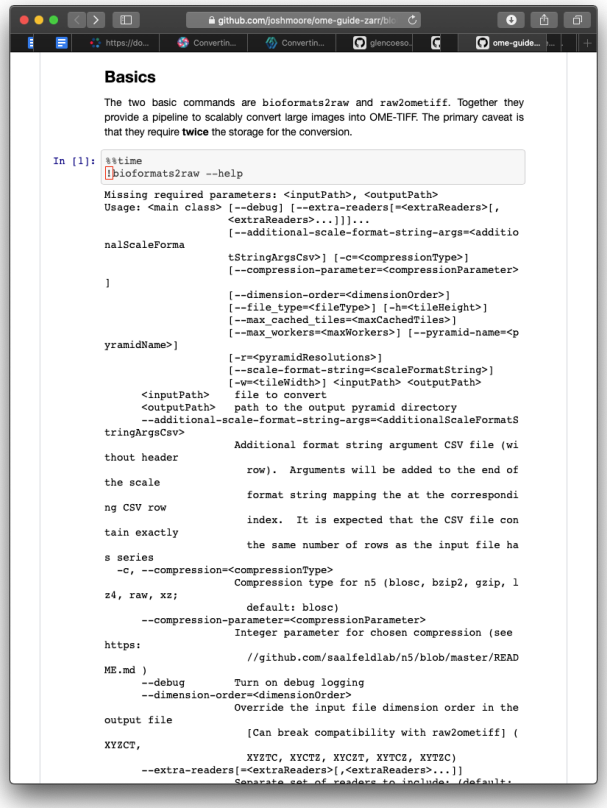


Slide 8 from https://downloads.openmicroscopy.org/presentations/2020/Dundee/main-talks/OME2020_NGFF_Moore.pdf

Links used during this section:

- https://forum.image.sc/t/converting-whole-slide-images-to-ome-tiff-a-new-workflow/32110
- https://www.glencoesoftware.com/blog/2019/12/09/converting-whole-slide-images-to-OME-TIFF.html
- https://github.com/glencoesoftware/bioformats2raw/releases
- https://github.com/glencoesoftware/raw2ometiff/releases
- https://anaconda.org/ome/bioformats2raw
- https://github.com/joshmoore/ome-guide-zarr/blob/master/notebooks/conversion.ipynb

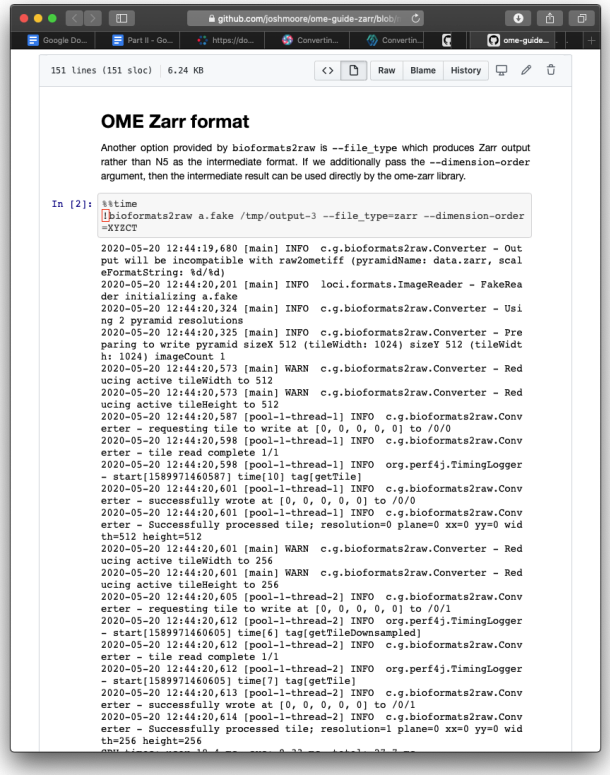




Basics

The two basic commands are `bioformats2raw` and `raw2ometiff`. Together they provide a pipeline to scalably convert large images into OME-TIFF. The primary caveat is that they require **twice** the storage for the conversion.

```
In [1]: %%time
        !bioformats2raw --help

Missing required parameters: <inputPath>, <outputPath>
Usage: <main class> [--debug] [--extra-readers[=<extraReaders>[,
                    <extraReaders>...]]]...
                    [--additional-scale-format-string-args=<additio
nalScaleForma
                    tStringArgsCsv>] [-c=<compressionType>]
                    [--compression-parameter=<compressionParameter>
]
                    [--dimension-order=<dimensionOrder>]
                    [--file_type=<fileType>] [-h=<tileHeight>]
                    [--max_cached_tiles=<maxCachedTiles>]
                    [--max_workers=<maxWorkers>] [--pyramid-name=<p
yramidName>]
                    [-r=<pyramidResolutions>]
                    [--scale-format-string=<scaleFormatString>]
                    [-w=<tileWidth>] <inputPath> <outputPath>
      <inputPath>    file to convert
      <outputPath>   path to the output pyramid directory
      --additional-scale-format-string-args=<additionalScaleFormatS
tringArgsCsv>
                     Additional format string argument CSV file (wi
thout header
                       row).  Arguments will be added to the end of
the scale
                       format string mapping the at the correspondi
ng CSV row
                       index.  It is expected that the CSV file con
tain exactly
                       the same number of rows as the input file ha
s series
  -c, --compression=<compressionType>
                     Compression type for n5 (blosc, bzip2, gzip, l
z4, raw, xz;
                       default: blosc)
      --compression-parameter=<compressionParameter>
                     Integer parameter for chosen compression (see
https:
                       //github.com/saalfeldlab/n5/blob/master/READ
ME.md )
      --debug        Turn on debug logging
      --dimension-order=<dimensionOrder>
                     Override the input file dimension order in the
output file
                       [Can break compatibility with raw2ometiff] (
XYZCT,
                       XYZTC, XYCZT, XYTCZ, XYTZC)
      --extra-readers[=<extraReaders>[,<extraReaders>...]]
                     Separate set of readers to include: (default:
```



151 lines (151 sloc) | 6.24 KB — Raw | Blame | History

OME Zarr format

Another option provided by `bioformats2raw` is `--file_type` which produces Zarr output rather than N5 as the intermediate format. If we additionally pass the `--dimension-order` argument, then the intermediate result can be used directly by the ome-zarr library.

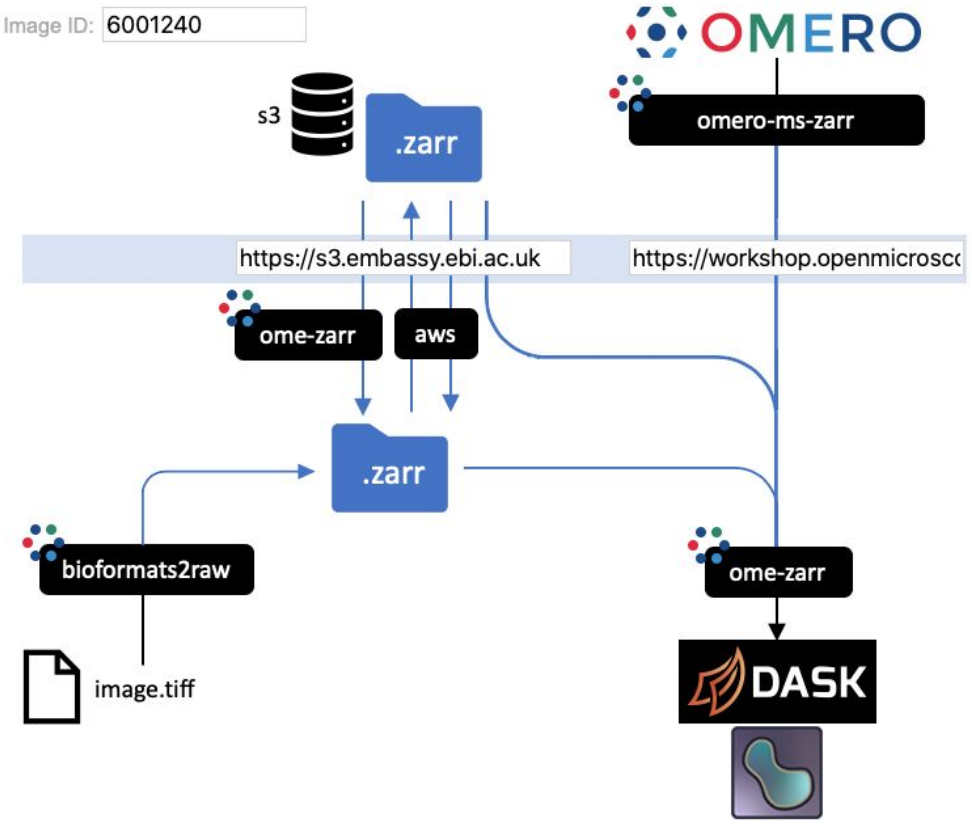
```
In [2]: %%time
        !bioformats2raw a.fake /tmp/output-3 --file_type=zarr --dimension-order
        =XYZCT

2020-05-20 12:44:19,680 [main] INFO  c.g.bioformats2raw.Converter - Out
put will be incompatible with raw2ometiff (pyramidName: data.zarr, scal
eFormatString: %d/%d)
2020-05-20 12:44:20,201 [main] INFO  loci.formats.ImageReader - FakeRea
der initializing a.fake
2020-05-20 12:44:20,324 [main] INFO  c.g.bioformats2raw.Converter - Usi
ng 2 pyramid resolutions
2020-05-20 12:44:20,325 [main] INFO  c.g.bioformats2raw.Converter - Pre
paring to write pyramid sizeX 512 (tileWidth: 1024) sizeY 512 (tileWidt
h: 1024) imageCount 1
2020-05-20 12:44:20,573 [main] WARN  c.g.bioformats2raw.Converter - Red
ucing active tileWidth to 512
2020-05-20 12:44:20,573 [main] WARN  c.g.bioformats2raw.Converter - Red
ucing active tileHeight to 512
2020-05-20 12:44:20,587 [pool-1-thread-1] INFO  c.g.bioformats2raw.Conv
erter - requesting tile to write at [0, 0, 0, 0, 0] to /0/0
2020-05-20 12:44:20,598 [pool-1-thread-1] INFO  c.g.bioformats2raw.Conv
erter - tile read complete 1/1
2020-05-20 12:44:20,598 [pool-1-thread-1] INFO  org.perf4j.TimingLogger
- start[1589971460587] time[10] tag[getTile]
2020-05-20 12:44:20,601 [pool-1-thread-1] INFO  c.g.bioformats2raw.Conv
erter - successfully wrote at [0, 0, 0, 0, 0] to /0/0
2020-05-20 12:44:20,601 [pool-1-thread-1] INFO  c.g.bioformats2raw.Conv
erter - Successfully processed tile; resolution=0 plane=0 xx=0 yy=0 wid
th=512 height=512
2020-05-20 12:44:20,601 [main] WARN  c.g.bioformats2raw.Converter - Red
ucing active tileWidth to 256
2020-05-20 12:44:20,601 [main] WARN  c.g.bioformats2raw.Converter - Red
ucing active tileHeight to 256
2020-05-20 12:44:20,605 [pool-1-thread-2] INFO  c.g.bioformats2raw.Conv
erter - requesting tile to write at [0, 0, 0, 0, 0] to /0/1
2020-05-20 12:44:20,612 [pool-1-thread-2] INFO  org.perf4j.TimingLogger
- start[1589971460605] time[6] tag[getTileDownsampled]
2020-05-20 12:44:20,612 [pool-1-thread-2] INFO  c.g.bioformats2raw.Conv
erter - tile read complete 1/1
2020-05-20 12:44:20,612 [pool-1-thread-2] INFO  org.perf4j.TimingLogger
- start[1589971460605] time[7] tag[getTile]
2020-05-20 12:44:20,613 [pool-1-thread-2] INFO  c.g.bioformats2raw.Conv
erter - successfully wrote at [0, 0, 0, 0, 0] to /0/1
2020-05-20 12:44:20,614 [pool-1-thread-2] INFO  c.g.bioformats2raw.Conv
erter - Successfully processed tile; resolution=1 plane=0 xx=0 yy=0 wid
th=256 height=256
```

Conversion examples (launch)	OME Zarr examples (launch)

Using the raw (i.e. Zarr) format for public data

Links used during this section:

- https://github.com/ome/omero-guide-python/blob/master/notebooks/zarr-public-s3-multiscale.ipynb
- https://github.com/joshmoore/ome-guide-zarr/blob/master/notebooks/zarr.ipynb
- https://downloads.openmicroscopy.org/presentations/2020/Dundee/zarr_diagram/
- e.g. ome_zarr info https://s3.embassy.ebi.ac.uk/idr/zarr/v0.1/6001240.zarr
- e.g. napari https://s3.embassy.ebi.ac.uk/idr/zarr/v0.1/9822151.zarr



Access workflows (diagram)

```
$ ome_zarr download https://s3.embassy.ebi.ac.uk/idr/zarr/v0.1/6001240.zarr
$ ome_zarr info 6001240.zarr

{'multiscales': [{'datasets': [{'path': '0'}, {'path': '1'}], 'version': '0.1'}],   'omero':
{'channels': [{'active': True, 'coefficient': 1, 'color': '0000FF', 'family': 'linear',
'inverted': False, 'label': 'LaminB1', 'window': {'end': 1500, 'max': 65535, 'min': 0,
'start': 0}}, {'active': True, 'coefficient': 1, 'color': 'FFFF00', 'family': 'linear',
'inverted': False, 'label': 'Dapi', 'window': {'end': 1500, 'max': 65535, 'min': 0, 'start':
0}}], 'id': 6001240, 'name': 'B1_C1.tif', 'rdefs': {'defaultT': 0, 'defaultZ': 118, 'model':
'color'}, 'version': '0.1'}}

...
```

(* If you plan on logging in: OMERO.server "workshop" credentials)

Binary layout of an image

Each image is currently stored as a separate Zarr fileset named according to its OMERO ID:

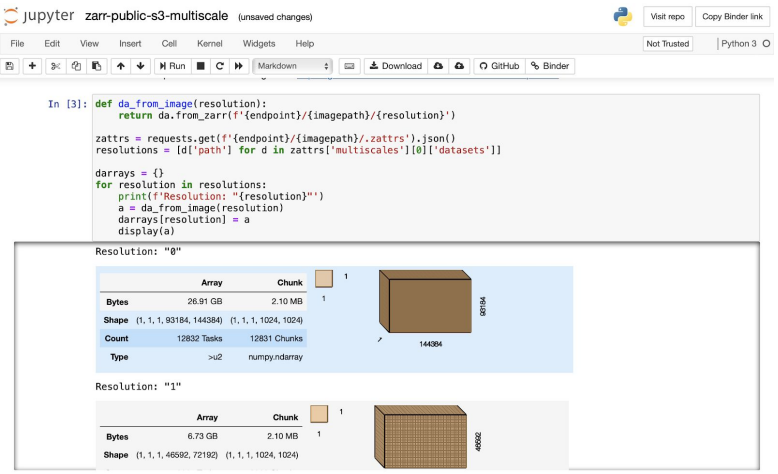
```
.                           # Root folder, potentially in S3,
├── 123.zarr                  # with a flat list of images by image ID.
└── 456.zarr                  #
    ├── .zattrs               # Group level metadata, including OMERO metadata.
    ├── .zgroup               # Each image is a Zarr group with multscale metadata.
    └── 0                     # Each multiscale level is stored as a separate Zarr array.
        ├── .zarray           #
        ├── 0.0.0.0.0         # Chunks are stored with the flat directory layout.
        └── t.c.z.y.x         # All image arrays are 5-dimensional
                            # with dimension order (t, c, z, y, x).
```

OMERO metadata

The OMERO metadata providing rendering settings and channel annotations is available under the “omero” key within the Zarr attributes (.zattrs) of the top-level group.

```
"id": 1,                            # ID in OMERO
"name": "example.tif",              # Name as shown in the UI
"channels": [                       # Array matching the c dimension size
    {
        "active": true,
        "color": "0000FF",
        "label": "LaminB1",
        "window": {
            "end": 1500,
            "max": 65535,
            "min": 0,
            "start": 0
        }...
    }
],
"rdefs": {
    "defaultT": 0,                  # First time point to show the user
    "defaultZ": 118,                # First Z section to show the user
    "model": "color"                # "color" or "greyscale"
}
```

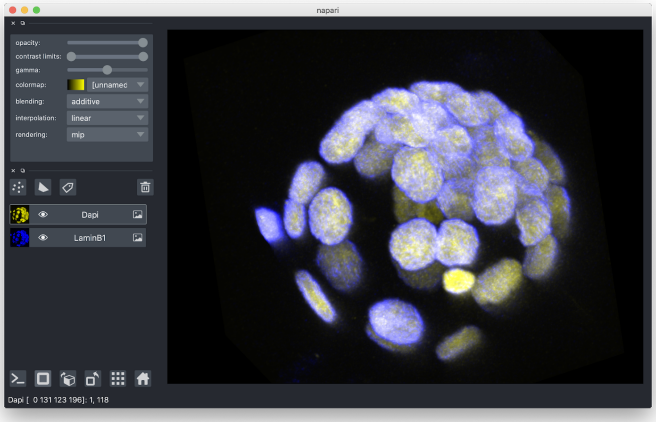
A notebook is available on mybinder.org to further explain the layout of the file format: [zarr-public-s3-multiscale.ipynb]

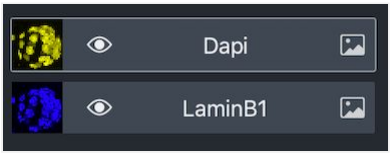


ome-zarr plugin for napari plugin

Links used during this section:

- https://github.com/ome/ome-zarr-py
- http://pypi.org/project/ome-zarr/






```
$ pip install napari==0.3.1 ome-zarr==0.0.3   # numcodecs may need to come from conda

$ napari https://s3.embassy.ebi.ac.uk/idr/zarr/v0.1/6001240.zarr

# or in Python:
with napari.gui_qt():
   viewer = napari.Viewer()
   viewer.open('https://s3.embassy.ebi.ac.uk/idr/zarr/v0.1/6001240.zarr/')
```