



OME Files C++ status

Roger Leigh

May 2017

Progress over the last year and upcoming features

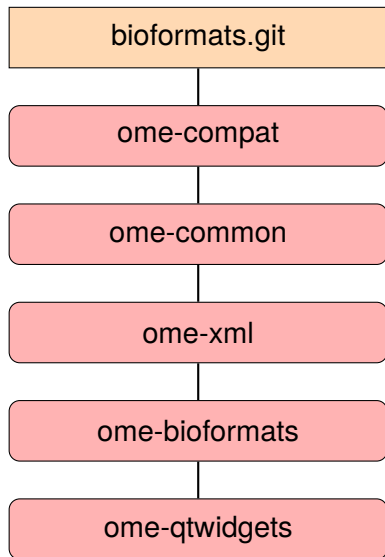


What *is* OME Files?

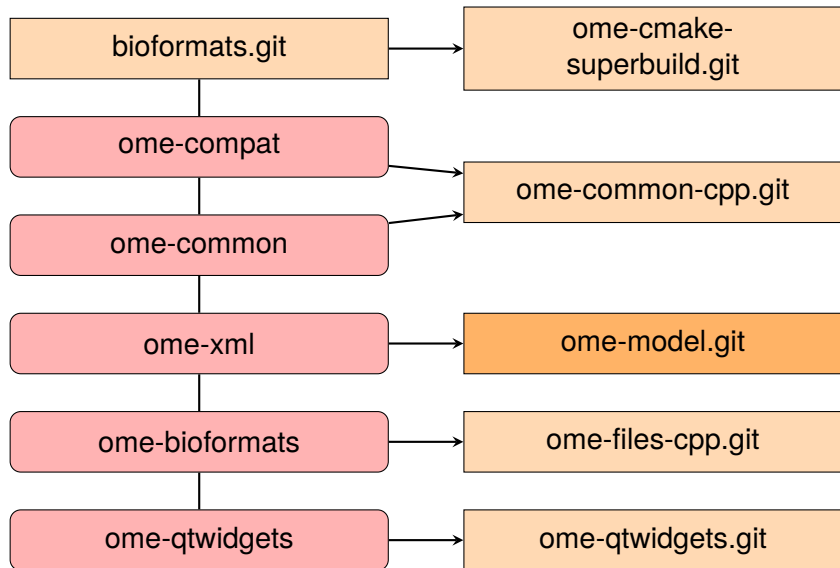
OME Files is a reference implementation of the OME Metadata Model and the OME-TIFF file format.

- ▶ Broad API compatibility with Bio-Formats Java
- ▶ Read and write support for OME-TIFF and TIFF
- ▶ OME Files: reader and writer API
- ▶ OME Model: model and metadata API

Component splitting: completed



Component splitting: completed



- ▶ Profiling of C++, Java and JNI (JACE)
- ▶ Real-world data-sets:
 - ▶ Simple 5D volume (small, limited metadata)
 - ▶ Plate (many images, large pixeldata size)
 - ▶ ROI (many ROIs, large metadata size)
- ▶ Detailed investigation with call and cache profiling (valgrind with callgrind and cachegrind)
- ▶ Ongoing work: tiling performance

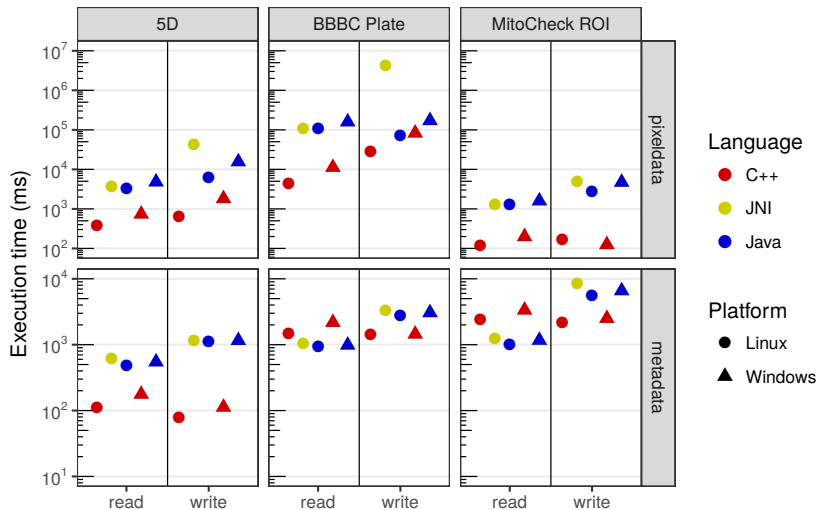
Repository: https://github.com/openmicroscopy/ome-files-performance/tree/0_1

Preprint:

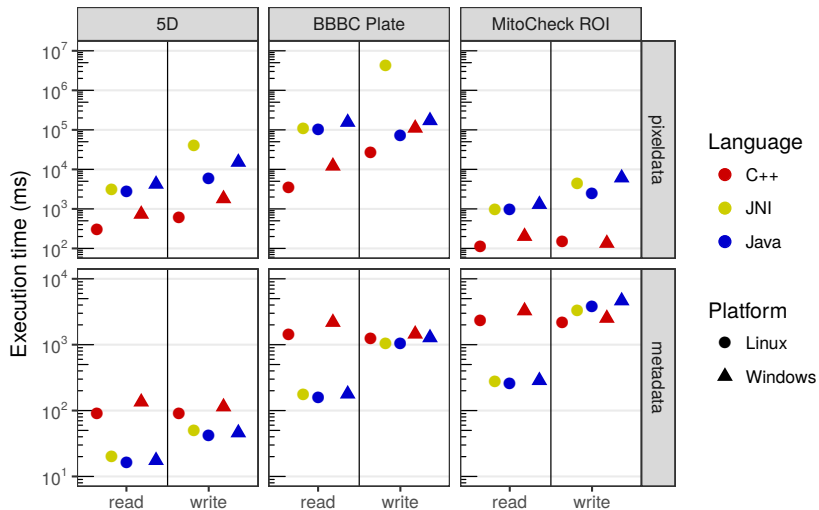
<http://biorxiv.org/content/early/2017/03/09/088740>

Zenodo: <https://zenodo.org/record/559270>

Performance testing results



Performance testing results (repeated)



- ▶ Added support for VS2015
- ▶ Dropped support for VS2012
- ▶ Added command line tools
- ▶ Numerous fixes and enhancements

Major features:

- ▶ Added 2015-01 and 2016-06 model support

Major features:

- ▶ Added 2015-01 and 2016-06 model support
- ▶ Units
- ▶ Map annotations
- ▶ Model has feature and API parity in C++ and Java
- ▶ Native unit and quantity support
- ▶ C++ and Java model maintained together in a unified codebase

Major features:

- ▶ Added 2015-01 and 2016-06 model support
- ▶ Units
- ▶ Map annotations
- ▶ Model has feature and API parity in C++ and Java
- ▶ Native unit and quantity support
- ▶ C++ and Java model maintained together in a unified codebase

Minor features:

- ▶ BinData
- ▶ ROI transforms

TIFF readers and writers

- ▶ added TIFF compression
- ▶ added default strip and tile size heuristic
- ▶ added IFD offset caching
- ▶ added OME-TIFF validity checks, metadata caching, file caching, plane element handling

```
writer.setCompression("lzw");  
writer.setInterleaved(false);  
writer.setTileSizeX(512U);  
writer.setTileSizeY(512U);
```

Additional changes

- ▶ C++11/14
- ▶ Native unit and quantity support
- ▶ Updated examples

Additional changes

- ▶ C++11/14
- ▶ Native unit and quantity support
- ▶ Updated examples

```
typedef model::primitives::Quantity
    <model::enums::UnitsLength,
        model::primitives::PositiveFloat> PositiveLength;
metadatastore.setPixelsPhysicalSizeX
    (PositiveLength(118.2,
        model::enums::UnitsLength::MICROMETER), 0);

// C++11
metadatastore.setPixelsPhysicalSizeX
    ({118.2, UnitsLength::MICROMETER}, 0);

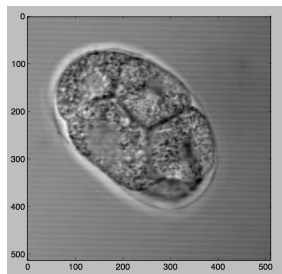
// Low level Boost.Units
micrometre_quantity len(118.2);
```

Python API

- ▶ <https://github.com/ome/ome-files-py>
- ▶ Core: Python extension module that wraps the C++ API
- ▶ Can open image planes as NumPy arrays
- ▶ Work in progress: exposes a subset of OMETIFFReader

```
import ome_files
import matplotlib.pyplot as plt

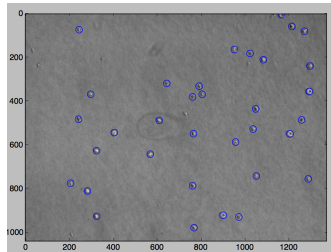
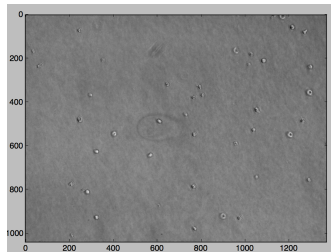
reader = ome_files.OMETIFFReader()
reader.set_id("tubhiswt_CO.ome.tif")
pixels = reader.open_array(0)
reader.close()
plt.imshow(pixels, cmap="gray")
plt.show()
```



Python API: external metadata

```
import matplotlib.pyplot as plt
import pandas as pd
import datapackage
import ome_files
import ome_files.metadata as ofmd

reader = ome_files.OMETIFFReader()
reader.set_id("9I5TT808_F00000010.companion.ome")
pixels = reader.open_array(0)
plt.imshow(pixels, cmap="gray")
plt.show()
meta = ofmd.OMEXMLMetadata(reader.get_ome_xml())
reader.close()
cmso_annotations = [
    _ for _ in meta.get_map_annotations()
    if _.Namespace == "CMSO/dpkg"
]
ann = cmso_annotations[0]
paths = ann.Value.get("FilePath")
dp = datapackage.DataPackage(paths[0])
res_map = dict((_.descriptor["name"],
    _.local_data_path) for _ in dp.resources)
df = pd.read_csv(res_map["objects_table"])
df = df[df["time index"] == 0]
x, y = df["cell col"].values, df["cell row"].values
plt.scatter(x=x, y=y, s=100, edgecolors='b',
    facecolors='none')
plt.imshow(pixels, cmap="gray")
plt.show()
```



- ▶ Flexible model annotations
 - ▶ Modulo
 - ▶ Direct serialisation of custom annotations

- ▶ Flexible model annotations
 - ▶ Modulo
 - ▶ Direct serialisation of custom annotations
- ▶ Options API (YAML?)

Feedback appreciated:

- ▶ Dropping support for VS2013 (deprecate then EOL?)
- ▶ Adding support for VS2017?
- ▶ Windows DLLs

Feedback requested (and help required)

Feedback appreciated:

- ▶ Dropping support for VS2013 (deprecate then EOL?)
- ▶ Adding support for VS2017?
- ▶ Windows DLLs

Help or advice needed:

- ▶ Supporting Windows DLLs

OME Files C++ now supports:

- ▶ The latest OME-TIFF and OME Model versions
- ▶ Complete API parity with Bio-Formats (modulo `Modulo`)

Performance profiling shows:

- ▶ Pixel data performance exceeds Java performance
- ▶ Metadata writing exceeds Java performance
- ▶ Metadata reading mostly exceeds Java performance