Supplementary Information

OME Files - An open source reference library for the OME-XML metadata model and the OME-TIFF file format

Leigh et al

This Supplementary information describes the set of benchmarks used to test the performance of the C++-based OME Files library. The scripts used for the benchmark tests are available at https://github.com/openmicroscopy/ome-files-performance.

Benchmark datasets

Three public reference OME-TIFF datasets were used for performance tests. For each dataset, we computed the metadata size-- the size in bytes of the raw OME-XML string stored in the ImageDescription TIFF tag-- and the pixeldata size-- the size in bytes of the binary pixel data stored as TIFF. The test datasets are:

- "5D", a multi-dimensional fluorescence image with 10 Z-sections, 2 channels, 43 timepoints available at https://downloads.openmicroscopy.org/images/OME-TIFF/2016-06/tubhiswt-4D/. The metadata size is 176KiB and the size of the pixeldata is 216MiB.
- "Plate", a plate containing 384 wells and 6 fields, derived from the Broad Bioimage Benchmark Collection resource [1] and available at https://downloads.openmicroscopy.org/images/OME-TIFF/2016-06/BBBC/. The metadata size is 2.3MiB and the size of the pixeldata is 3.4GiB.
- "ROI", a time-lapse sequence with ~13K regions of interest, derived from the MitoCheck study [2] and available at https://downloads.openmicroscopy.org/images/OME-TIFF/2016-06/MitoCheck/. The metadata size is 3.2MiB and the size of the pixeldata is 130MiB.

The datasets were chosen to test different aspects of library performance. The Plate and ROI datasets are both single OME-TIFF derived from real-world examples where the file content is either dominated by the pixeldata or the metadata. 5D represents file layouts where the pixeldata is distributed over multiple files. For more information, see https://www.openmicroscopy.org/site/support/ome-model/ome-tiff/data.html.

[1] Ljosa V, Sokolnicki KL, Carpenter AE (2012). Annotated high-throughput microscopy image sets for validation. Nature Methods 9(7):637.

[2] Neumann B et al. (2010). Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. Nature 464(7289):721.

Benchmark hardware and software

The benchmark scripts have been executed using two identical sleds in a single server provisioned with the same hardware (Dell PowerEdge™ C6220, 2x E5-2640/96 GB). Windows 2008 Server was installed on one sled, Ubuntu 16.04 running in a Docker virtualized environment was installed on the other.

Our benchmark tests measured the performance of the current versions of our two reference libraries: OME Files C++ 0.3.1 and Bio-Formats version 5.3.4. The Bio-Formats library was executed directly using the Java Virtual Machine but also over C++ using the Bio-Formats JNI/JACE C++ bindings. All benchmark tests have been executed under both Windows and Linux environments with the exception of the JNI/JACE C++ bindings which were only successfully built under Ubuntu 16.04. We used Java Development Kit (1.7.0_80 on Windows and 1.7.0_95 on Linux) to build and run the benchmark scripts as JDK7 is currently required to build the Bio-Formats JACE C++ bindings.

Benchmark tests

For each of the benchmark datasets above, four tests were executed:

- metadata.read: the metadata is extracted from the OME-TIFF ImageDescription tag and converted into OME Data Model objects using the createOMEXMLMetadata API (Java [3] / C++ [4])
- metadata.write: the metadata is serialized using the getOMEXML API (Java [5] / C++ [6]) and written to disk as an OME-XML file
- pixeldata.read: the pixeldata is read from the OME-TIFF using the openBytes API (Java [7] / C++ [8]) and stored in memory
- pixeldata.write: the pixeldata is written to disk as another OME-TIFF using the saveBytes API (Java [9] / C++ [10])

Each benchmark test records the real time in milliseconds before and after each test, and computes the elapsed time from the difference.

[3] https://downloads.openmicroscopy.org/bio-formats/5.3.4/api/loci/formats/services/OMEXMLService.html#createOMEXMLMetadata-java.lang.String
[4] https://downloads.openmicroscopy.org/ome-files-cpp/0.3.1/21/docs/ome-files-bundle-docs-0.3.1-b21/ome-files/api/html/namespaceome_1_1files.html#a469d4ec5c1bddd7b3afc0daa11ba1989
[5] https://downloads.openmicroscopy.org/bio-formats/5.3.4/api/loci/formats/services/OMEXMLService.html#getOMEXML-loci.formats.meta.MetadataRetrieve-
[6] https://downloads.openmicroscopy.org/ome-files-cpp/0.3.1/21/docs/ome-files-bundle-docs-0.3.1-b21/ome-files/api/html/namespaceome_1_1files.html#ad2898e87098e67fdda2154d7883692e0
[7] https://downloads.openmicroscopy.org/bio-formats/5.3.4/api/loci/formats/IFormatReader.html#openBytes-int-byte:A-
[8] https://downloads.openmicroscopy.org/ome-files-cpp/0.3.1/21/docs/ome-files-bundle-docs-0.3.1-b21/ome-files/api/html/classome_1_1files_1_1detail_1_1FormatReader.html#a2106d1dd7b4f4fe6597fde5cdbdb0f37

[9] https://downloads.openmicroscopy.org/bio-formats/5.3.4/api/loci/formats/IFormatWriter.html#saveBytes-int-byte:A-

[10] https://downloads.openmicroscopy.org/ome-files-cpp/0.3.1/21/docs/ome-files-bundle-docs-0.3.1-b21/ome-files/api/html/classome_1_1files_1_1detail_1_1FormatWriter.html#a51115641c238f5830f796c1839d75872

Building and executing the benchmark scripts

Windows

The Windows build requirements are Cmake (https://cmake.org/), Maven (http://maven.apache.org/), Visual Studio (https://www.visualstudio.com/) and a local version of the standalone OME Files bundle matching the Visual Studio version. For running our builds, we used the Continuous Integration software Jenkins (https://jenkins.io/index.html) to trigger the Windows benchmark builds. A single script executing the building and execution steps is available under jenkins_build.bat (https://github.com/openmicroscopy/ome-files-performance/blob/v0.1.0/scripts/jenkins_build.bat).

To build the OME Files performance scripts manually, within a build directory, execute the following cmake command:

```
$ cmake -G "Ninja" -DCMAKE_VERBOSE_MAKEFILE:BOOL=%verbose%
  -DCMAKE_INSTALL_PREFIX:PATH=%installdir% -
DCMAKE_BUILD_TYPE=%build_type%
  -DCMAKE_PREFIX_PATH=%OME_FILES_BUNDLE%
  -DCMAKE_PROGRAM_PATH=%OME_FILES_BUNDLE%\bin
  -DCMAKE_LIBRARY_PATH=%OME_FILES_BUNDLE%\lib
  -DBOOST_ROOT=%OME_FILES_BUNDLE% %sourcedir%
$ cmake --build .
$ cmake --build . --target install
```

The Bio-Formats performance script can be built within the source directory using Maven:

```
$ cd source
$ call mvn clean install
```

Linux

The Linux benchmark was performed on Ubuntu 16.04. To ease the distribution and reproducibility of the suite, the benchmark environment is built using Docker (https://www.docker.com/) via a Dockerfile (https://github.com/openmicroscopy/ome-files-performance/blob/v0.1.0/Dockerfile). To build the benchmark Docker image, run:

$ docker build -t ome-files-performance .

In order to execute the benchmark scripts, download the benchmark datasets under a local folder, e.g. /tmp/benchmark_data, then mount this local folder as a /data volume and run the Docker image:

$ docker run --rm -it -v /data:/data ome-files-performance

This will execute the run_benchmarking (https://github.com/openmicroscopy/ome-files-performance/blob/v0.1.0/scripts/run_benchmarking) script and store the output of the benchmark under /data/out and the tabular results under /data/results.

Benchmark results

Each of the benchmark tests outputs a tabular-separated values file with the following columns:

- test.lang: name of the benchmark environment (Java, C++, Jace)
- test.name: name of the benchmark test
- test.file: name of the benchmark dataset
- proc.real/real: execution time measured by the benchmark script

The results folder of the GitHub source code repository contains the final benchmarking run used to generate Fig. 2 and Supp. Table 1.

From these tab-separated value files, the following metrics have been defined for the assessment of each benchmark test:

- performance is defined as the inverse of the execution time for each test,
- relative performance of a test is defined as the ratio of the performance over the performance of the same test for the same dataset executed using Bio-Formats under Linux or Windows, as appropriate,
- metadata rate i.e. the rate of XML transfer per unit of time expressed in MiB/s is defined as the ratio of the metadata size of the test dataset over the execution time of the metadata test,
- pixeldata rate i.e. the rate of binary pixeldata transfer per unit of time expressed in MiB/s is defined as the ratio of the the pixeldata size of the test dataset over the execution time of the pixeldata test.

The benchmark metrics have been derived from twenty independent iterations of each benchmark tests. The only exception is the Plate dataset pixeldata performance test which has only been reproduced 6 times as a result of its long execution time (~1.5hr per test).

Additionally, we have reproduced the benchmark by repeating the same number of iterations of each test in a loop within the same environment. For most tests, the results were found to be identical and independent of whether the tests were run separately or repeated. Interestingly, in the case of the metadata tests using Bio-Formats and the Java Virtual Machine (JVM), there is a gain due to the optimiser in the JVM. We include these results for completeness and to indicate the performance achieved if the same operation is invoked within a process multiple times.

Supplementary Table 1: Benchmarking results. Each row of the table contains the processed metadata and pixeldata performance results for a given test dataset under a given environment. Execution times, relative performance, metadata rate and pixeldata rate are calculated as defined in the Supplementary Information. The mean and standard deviation of each value has been computed from twenty independent iterations of the same benchmark execution, except for the Plate pixeldata test (see Supplementary Information).