MuscleX Reports

This document is regrouping all the tests made concerning the speed, coverage and memory usage of MuscleX. Tests have been made under Python 3.8 with MuscleX v1.21.

# Memory usage reports

**To use memory profiler:**

**mprof run musclex qf**

**To display the result:**

**mprof plot**

**To have the details, run:**

**python -m memory\_profiler example.py**

The image tested is 39MB (uncompressed):

data/EIGERtestdatacardiac/P2\_F5\_849\_1\_094\_data\_000001\_0001.tif

The H5 tested is 216MB :

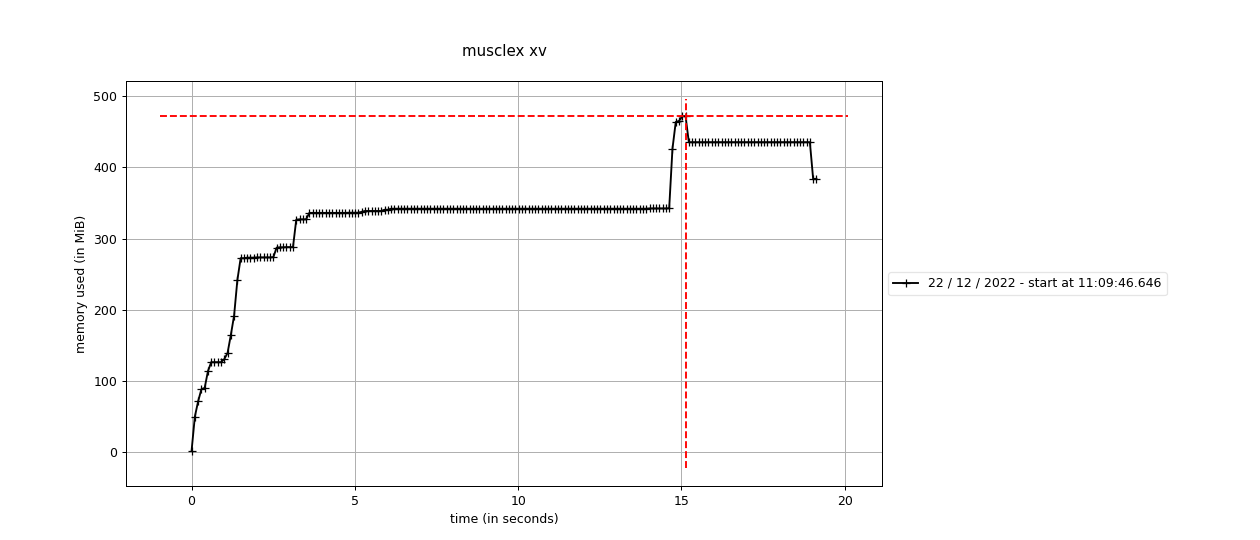
data/EIGERtestdatacardiac/h5/P2\_F5\_849\_1\_094\_data\_000001.h5

An H5 file is compressed, but its true size is 4000MB (h5dump -pH).

In the reports, the graphs are in MiB, and 1 MiB is equal to 1.05 MB.

## X-Ray Viewer

Tested on a TIF image:

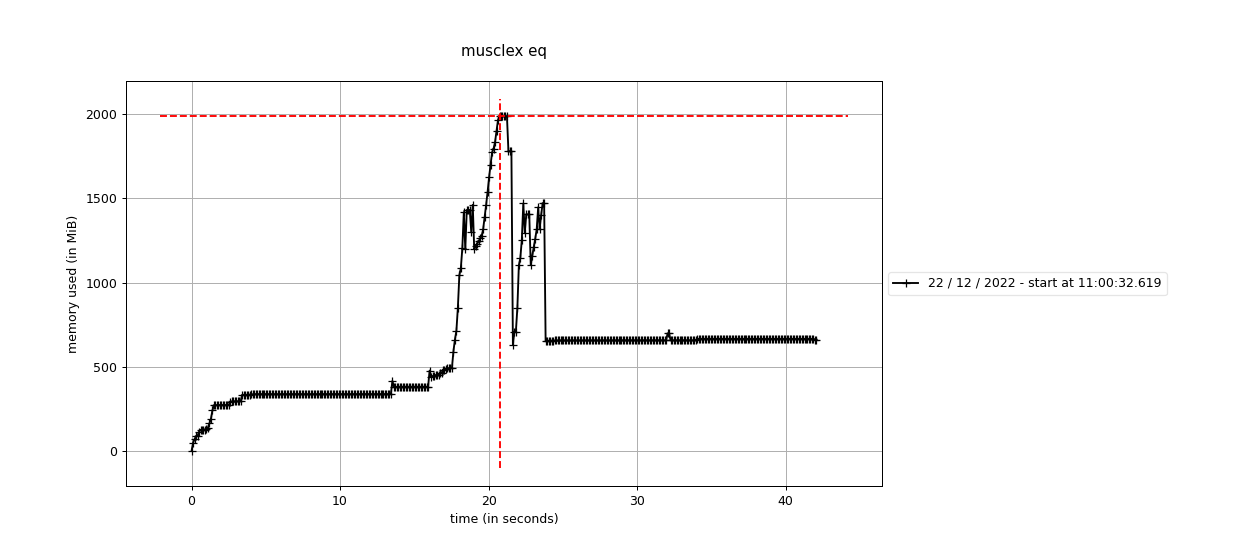


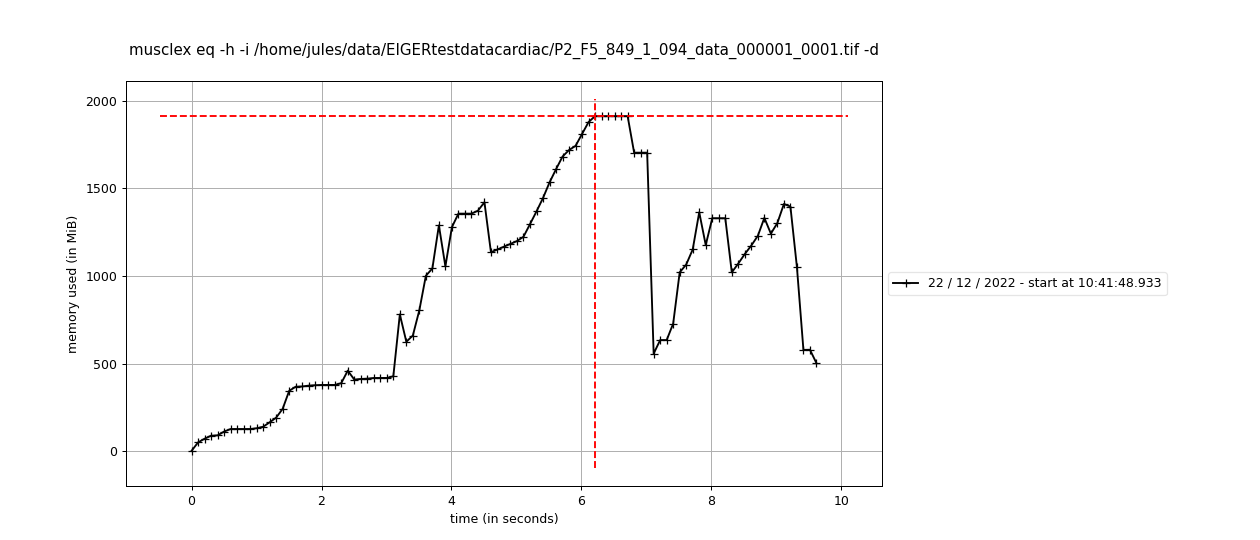
X-Ray viewer gives us a good idea of how much the program costs by itself (without any processing). The base usage is 380MB (Python, Qt5) and with the image (40MB).

## Equator

Memory multiplier = (2000 - 600)/40 = 35

The baseline with the added modules is 600MB (400MB before the other modules)

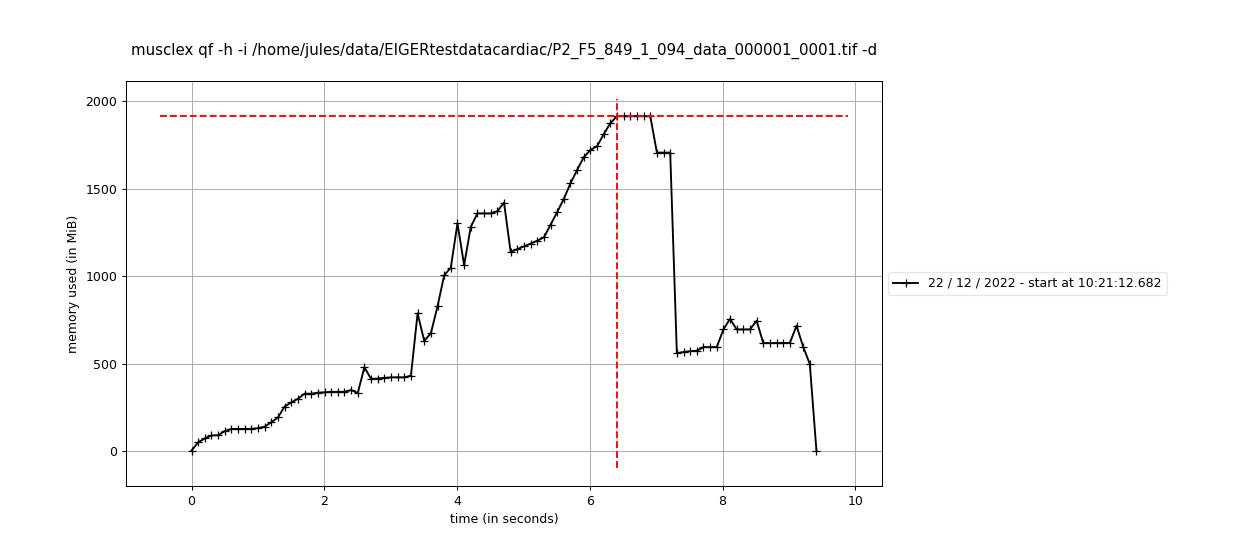
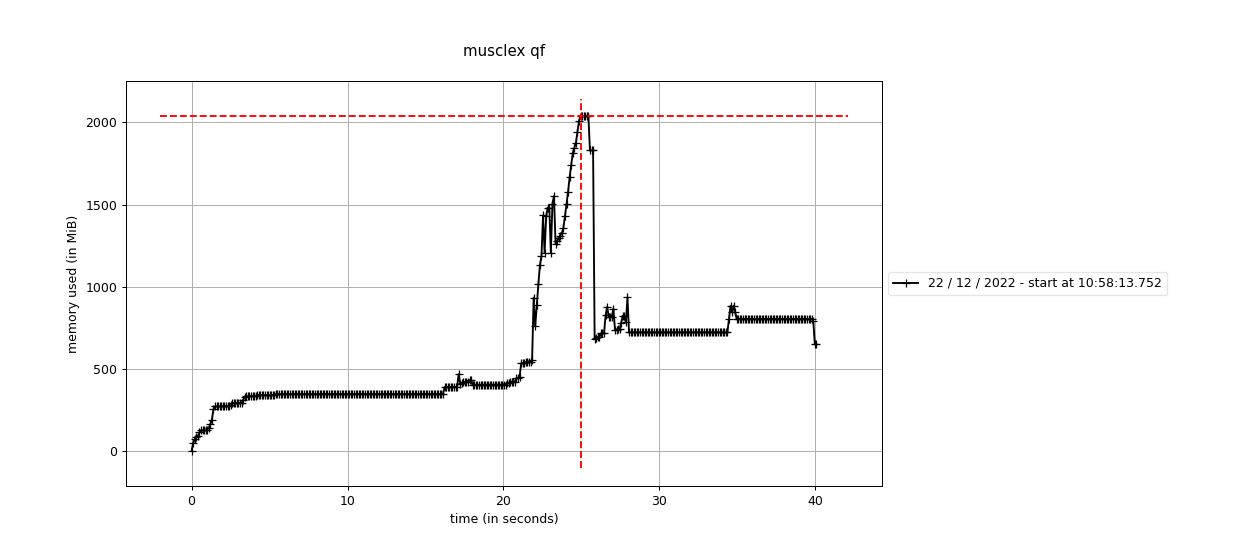




## 

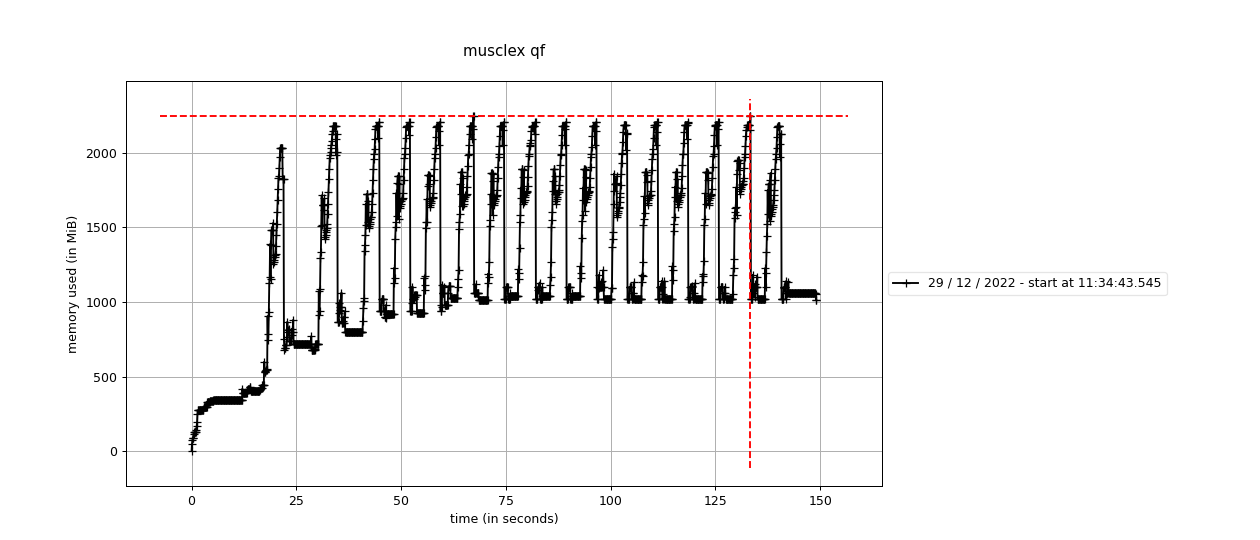
GUI (top) and Headless (bottom) give a similar result: around 2GB at the peak which is happening during rotationAngle.

## Quadrant Folder



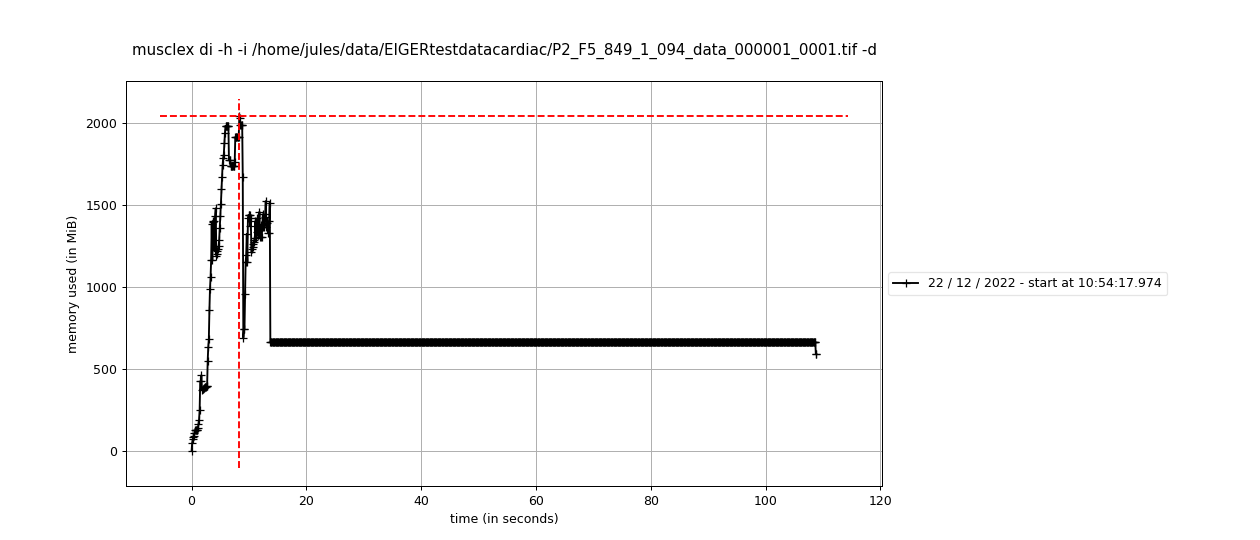
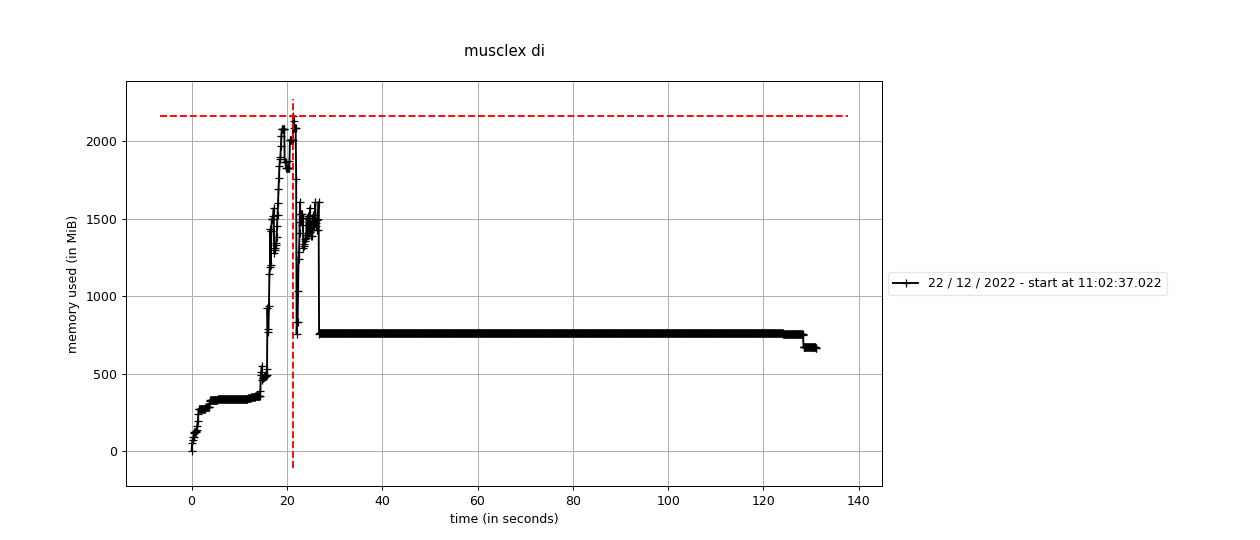
GUI (top) and Headless (bottom) give a similar result: around 2GB at the peak which is happening during rotationAngle.

Memory leak test on QF:



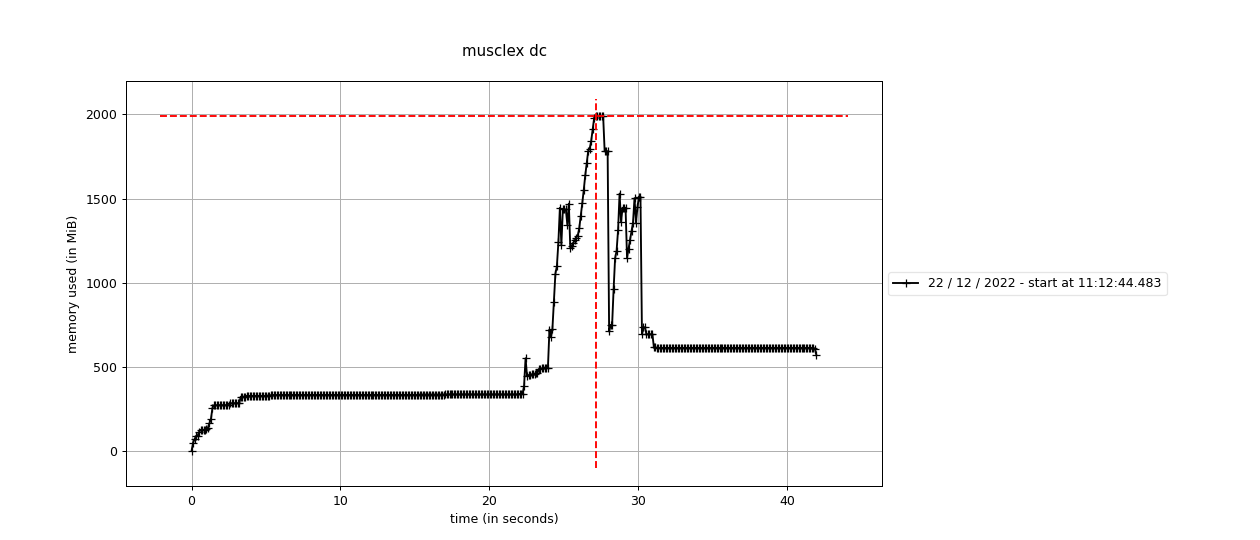
We can see that the memory is rising over the first 2 or 3 images before being stable.

## Scanning Diffraction



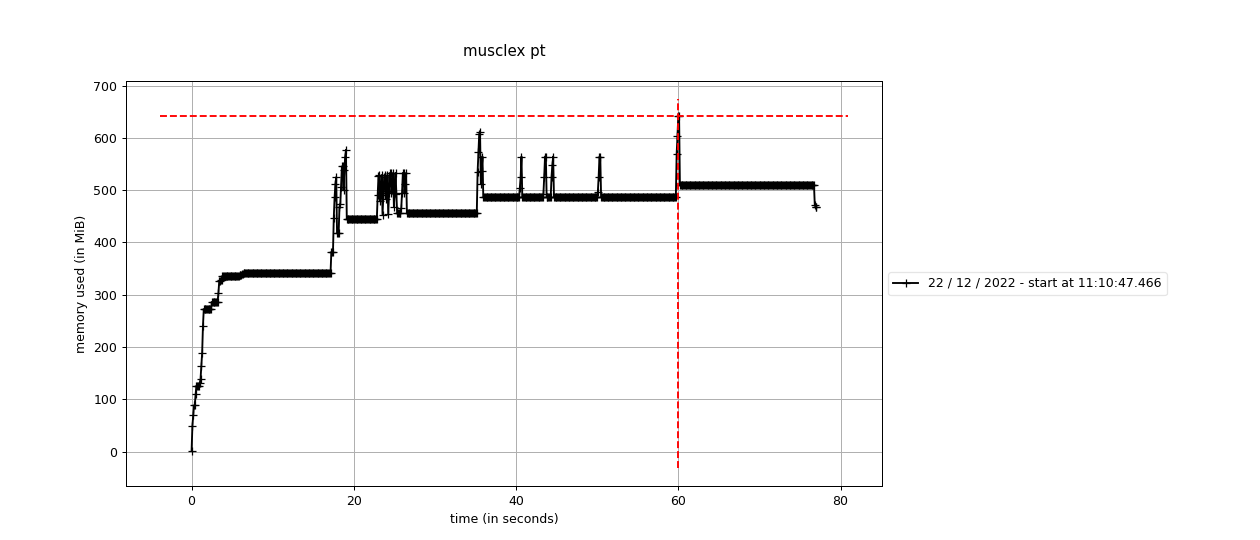
GUI (top) and Headless (bottom) give a similar result: around 2GB at the peak. There is a long and constant processing time in this function (more than 80 seconds). It is due to lmfit. Fixing the maximum number of function evaluations should fit this problem.

## Diffraction Centroids



Peak at around 2GB.

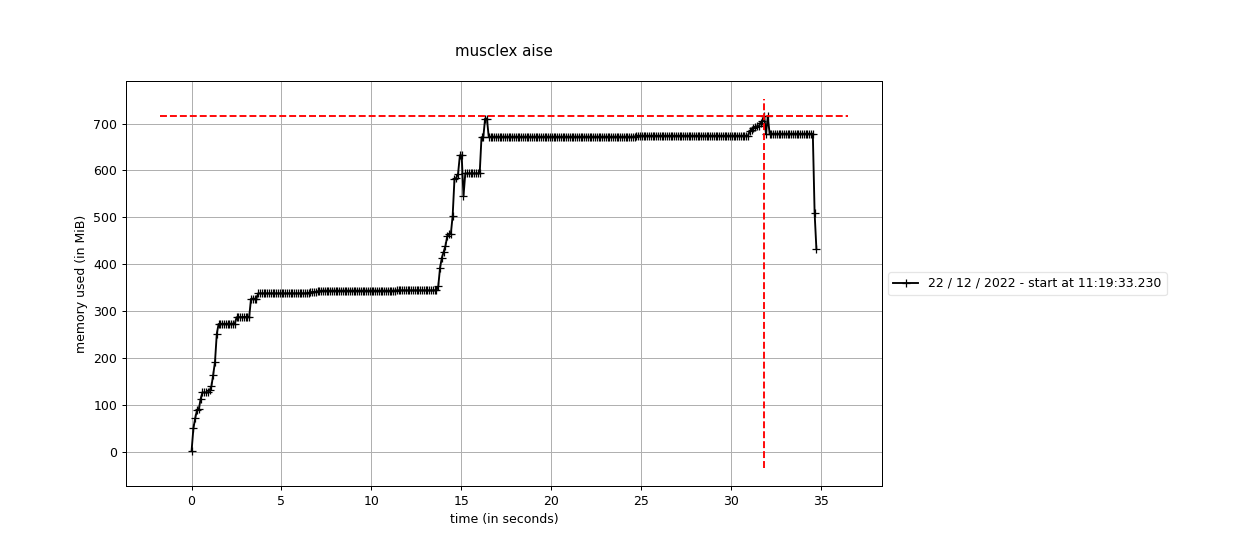
## Projection Traces



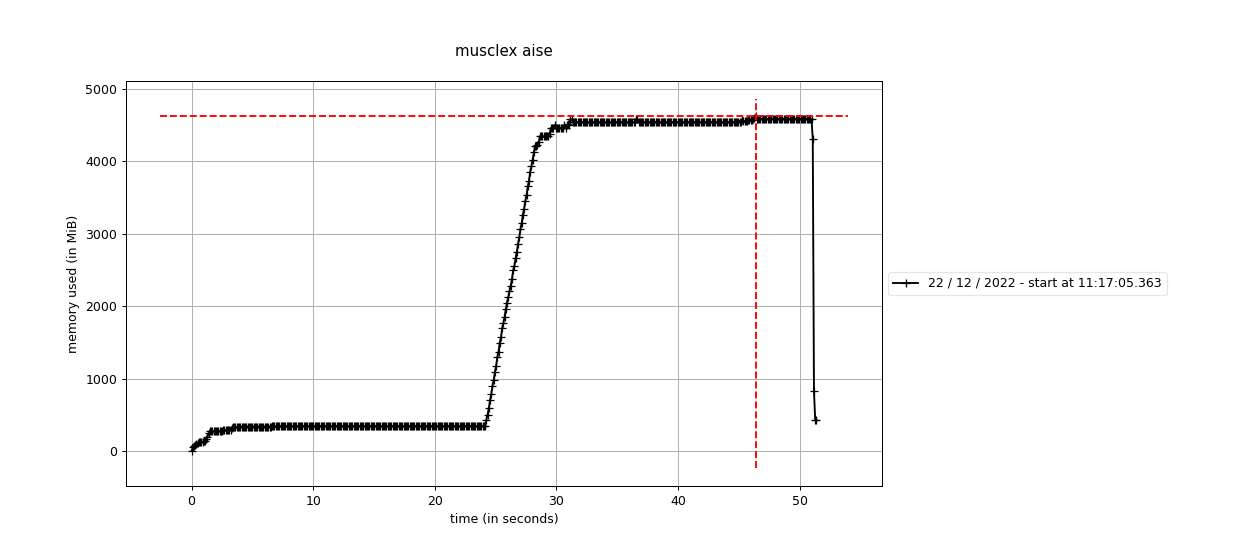
No noticeable peak.

## Add Intensities Single Experiment

AISE on a folder of TIFF images:



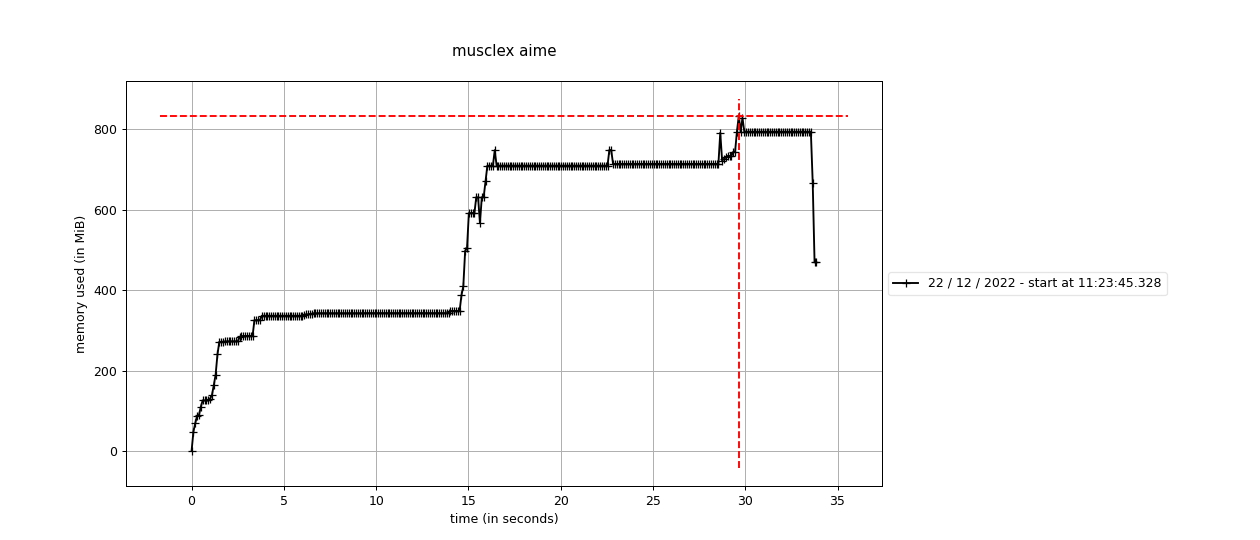
AISE on an h5 file:



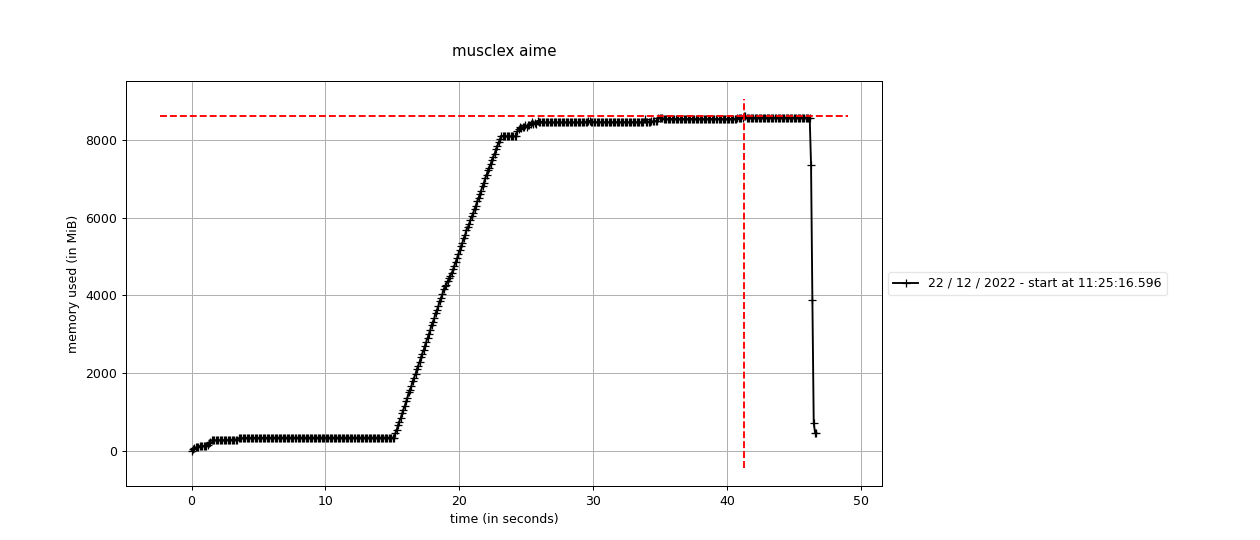
We can see that the h5 files make the process use a lot more memory. Indeed, opening an h5 file opens all the images inside of it, making it very memory consuming. Maybe we need to look at how to open a smaller group of images at the same time.

## Add Intensities Multiple Experiments

AIME on a folder of TIF images:



AIME on a folder of h5 files:



We can see that the h5 files make the process use a lot more memory. Indeed, opening an h5 file opens all the images inside of it, making it very memory consuming. Maybe we need to look at how to open a smaller group of images at the same time.

# Calls reports

**To profile calls, add in main.py by replacing what is in *if \_\_name\_\_ == “\_\_main\_\_”* by:**

import cProfile

cProfile.run('main(sys.argv)', 'output.dat')

import pstats

from pstats import SortKey

with open('output\_time.txt', 'w') as f:

p = pstats.Stats('output.dat', stream=f)

p.sort\_stats('time').print\_stats()

with open('output\_calls.txt', 'w') as f:

p = pstats.Stats('output.dat', stream=f)

p.sort\_stats('calls').print\_stats()

**Run the program:**

**python main.py di -h -i ~/data/EIGER\_testImages/17ER\_190\_data\_000001\_0001.tif -d**

This part was only tested on Headless programs (EQ, QF, DI).

## Equator Headless

Tue Jan 3 09:52:51 2023 output.dat

249273 function calls (243394 primitive calls) in 13.156 seconds

**Ordered by: internal time**

ncalls tottime percall cumtime percall filename:lineno(function)

2 4.296 2.148 8.474 4.237 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/pyFAI/azimuthalIntegrator.py:447(setup\_CSR)

2 1.732 0.866 1.732 0.866 {morphologyEx}

3 1.084 0.361 1.340 0.447 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/pyFAI/geometry.py:703(corner\_array)

697 0.925 0.001 0.925 0.001 {method 'reduce' of 'numpy.ufunc' objects}

**Ordered by: call count**

ncalls tottime percall cumtime percall filename:lineno(function)

25215/24933 0.002 0.000 0.003 0.000 {built-in method builtins.len}

20565/20535 0.003 0.000 0.004 0.000 {built-in method builtins.isinstance}

17018 0.002 0.000 0.002 0.000 {method 'append' of 'list' objects}

16031 0.004 0.000 0.005 0.000 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/lmfit/model.py:788(\_strip\_prefix)

The slow functions are part of pyFAI so there is not much to do to improve it.

## Quadrant Folder Headless

Tue Jan 3 09:54:20 2023 output.dat

2144340 function calls (2034956 primitive calls) in 12.461 seconds

**Ordered by: internal time**

ncalls tottime percall cumtime percall filename:lineno(function)

2 3.421 1.711 6.653 3.326 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/pyFAI/azimuthalIntegrator.py:447(setup\_CSR)

2 1.674 0.837 1.674 0.837 {morphologyEx}

2 0.952 0.476 0.952 0.476 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/pyFAI/geometry.py:965(deltaChi)

4 0.650 0.162 0.807 0.202 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/pyFAI/geometry.py:703(corner\_array)

**Ordered by: call count**

ncalls tottime percall cumtime percall filename:lineno(function)

311183/311156 0.033 0.000 0.045 0.000 {built-in method builtins.isinstance}

58719/57514 0.006 0.000 0.006 0.000 {built-in method builtins.len}

56686/56582 0.012 0.000 0.022 0.000 {built-in method builtins.getattr}

48356/8757 0.037 0.000 0.053 0.000 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/numba/core/ir.py:313(\_rec\_list\_vars)

The slow functions are part of pyFAI so there is not much to do to improve it.

## Scanning Diffraction Headless

Tue Jan 3 09:50:55 2023 output.dat

1225196 function calls (1190840 primitive calls) in 23.029 seconds

**Ordered by: internal time**

ncalls tottime percall cumtime percall filename:lineno(function)

10 6.705 0.670 12.305 1.230 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/pyFAI/azimuthalIntegrator.py:447(setup\_CSR)

10 2.981 0.298 9.318 0.932 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/pyFAI/azimuthalIntegrator.py:1099(integrate1d\_ng)

9 1.651 0.183 1.651 0.183 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/pyFAI/geometry.py:965(deltaChi)

1 1.435 1.435 1.474 1.474 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/musclex-1.21.0-py3.8.egg/musclex/modules/ScanningDiffraction.py:511(get\_runs\_from\_image)

**Ordered by: call count**

ncalls tottime percall cumtime percall filename:lineno(function)

174445/174257 0.012 0.000 0.012 0.000 {built-in method builtins.len}

126329 0.040 0.000 0.059 0.000 /home/jules/Documents/venv/mmusclex\_git/lib/python3.8/site-packages/lmfit/model.py:788(\_strip\_prefix)

108397/108367 0.012 0.000 0.015 0.000 {built-in method builtins.isinstance}

100291 0.012 0.000 0.012 0.000 {method 'startswith' of 'str' objects}

The slow functions are part of pyFAI so there is not much to do to improve it.

# Coverage reports

**To use coverage.py:**

**coverage run -m main test\_global**

**To generate the file:**

**coverage html -i**

This part gives a summary of the coverage of the whole program, but it is using “test\_global”, which is running consistency tests but only on Equator, Quadrant Folder and Scanning Diffraction.

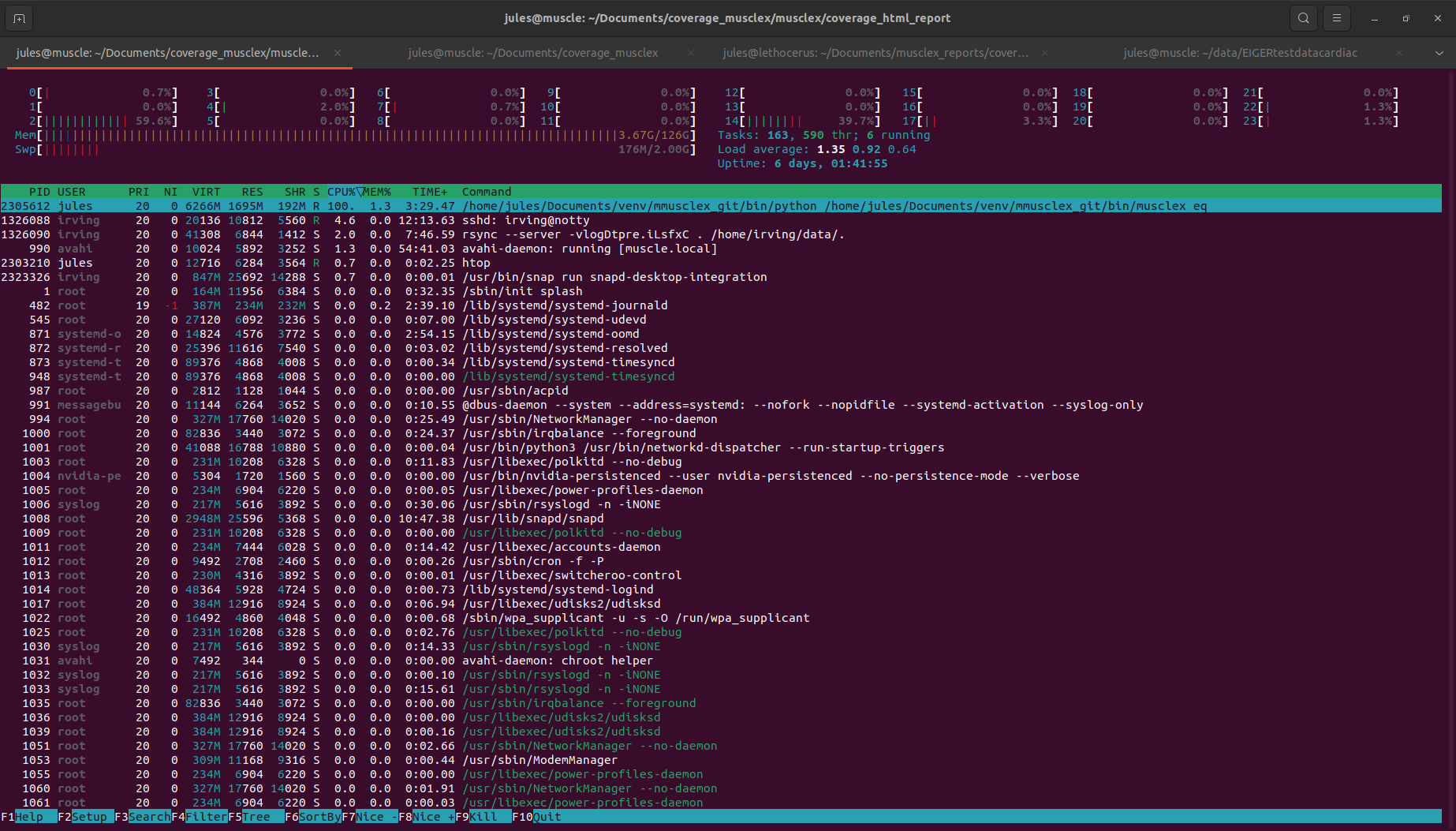
On lethocerus:

file:///home/jules/Documents/musclex\_reports/coverage\_musclex/musclex/coverage\_html\_report/index.html

# CPU Usage

Run: **htop**

Run MuscleX



100% of the CPU used when musclex eq is running for example.

*Jules N., 12/28/2022*