

Lazy and parallel bio-image processing using DASK (and python tools)

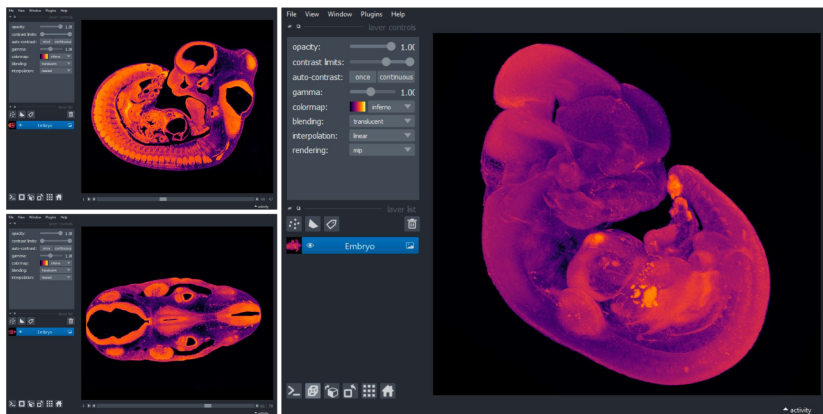
PoL BioImage Analysis Training School
30/8/2023
Marvin Albert

This course

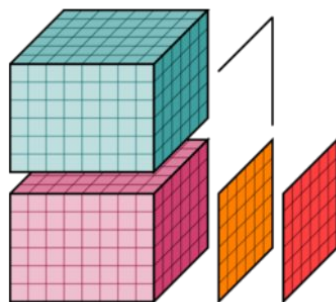
- Let's assume we use python software for bio-image analysis, e.g.
 - its scientific software stack
 - napari
- In this course we'll discuss hands-on ways to
 - deal with large image data
 - accelerate workflows (on a laptop, workstation, cluster or GPU)
- We'll focus around the use of DASK

Dask is widely used in community software

Napari

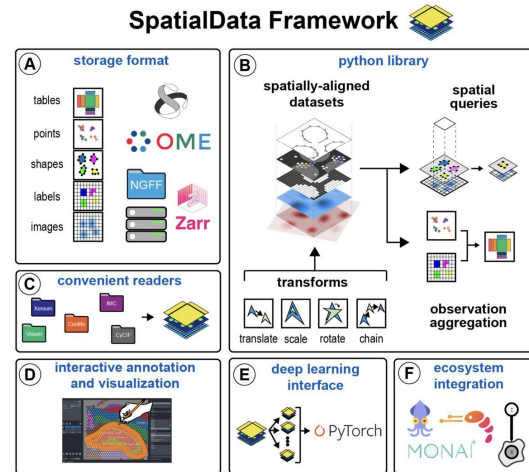


Xarray



scverse

SpatialData Framework

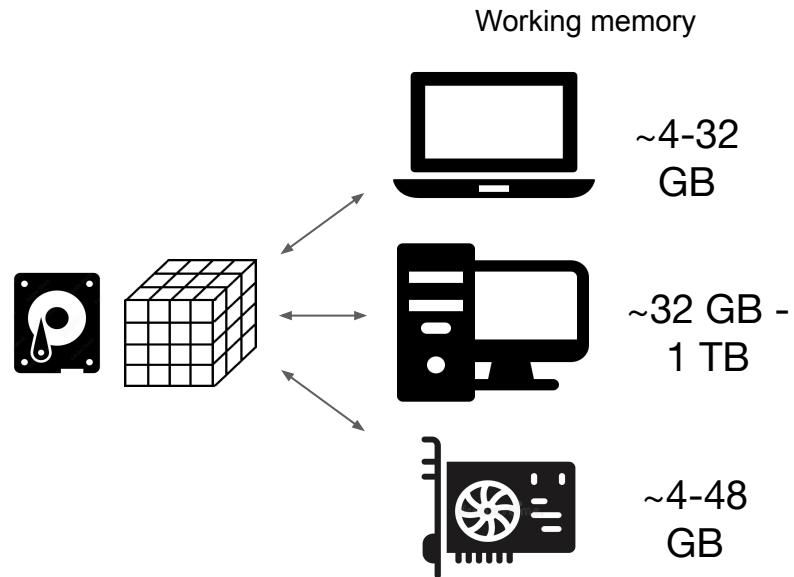


Challenge: large data

Image data sizes:

- $X * Y * Z * C * T$ bytes / pixel
- multiplied during processing

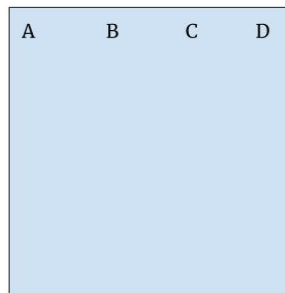
	X	Y	Z	C	T	Bytes/Px	Total
Light sheet (single stack)	2000	2000	500	1	1	2B (uint16)	4 GB
Light sheet (full dataset)	-	-	-	2	100	2B	800 GB
Digital Pathology & HCS	3000 0	3000 0	1	1	1	3B	3 GB
Volume EM	2000	2000	2000	1	1	2	16 GB



Chunked/lazy file storage formats

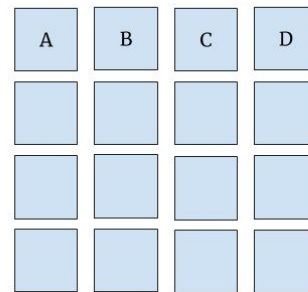
- Reading and writing of arrays in chunks
- Lazy data access: at the time needed
- Examples: HDF5, zarr, N5, netcdf, tif
- Useful for viewer
- But: How to process an array chunk by chunk?

Not Chunked



Bytes on disk
AABBCDDAAABBCDDAAABBCDDAAABB...

Chunked

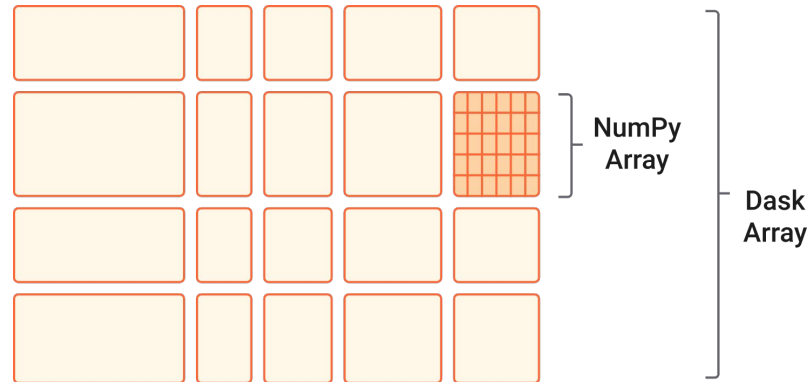


Bytes on disk
AAAAAAAABBBBBBBCCCCCCCCDDDDDD...

<https://napari.staging.imaging.cziscience.com/guides/stable/rendering-explanation.html>

Dask enables lazy and parallel execution of python code

- General idea: delayed execution of python functions
- Dask arrays:



What does *dask* do?

Collections

(create task graphs)

Dask Array

Dask DataFrame

Dask Bag

Dask Delayed

Futures

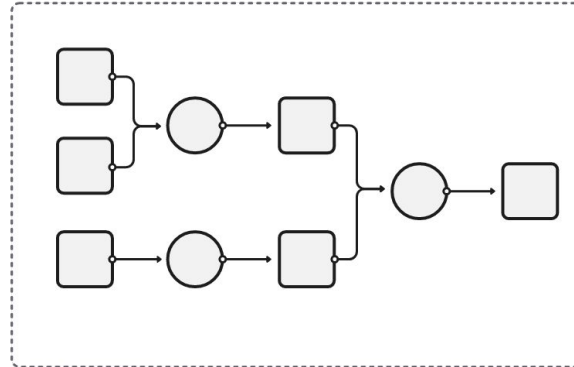


Task Graph



Schedulers

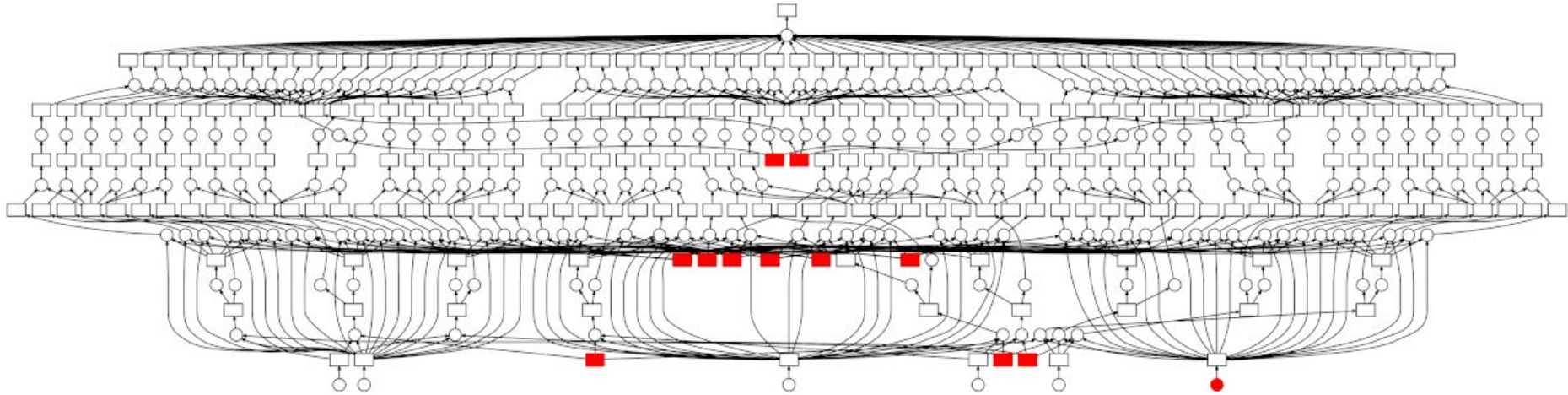
(execute task graphs)



Single-machine
(threads, processes,
synchronous)

Distributed

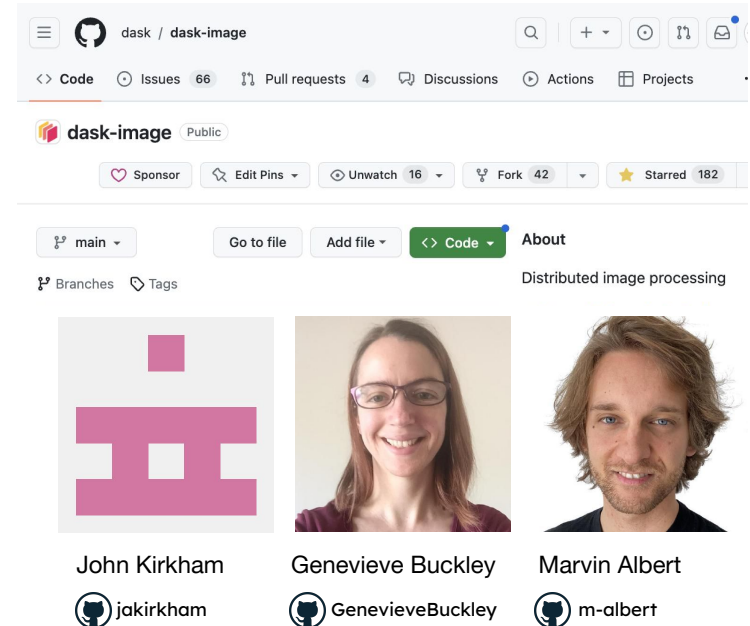
How does dask work?



Lazy and parallel image processing: dask-image

github.com/dask/dask-image

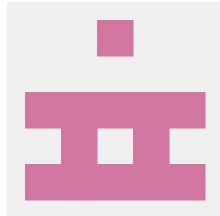

- enables parallel and chunked N-D image processing
- makes SciPy's ndimage functionality available for dask arrays
- includes CuPy support for GPU processing







main

Go to file Add file Code About

Branches Tags Distributed image processing

 John Kirkham
 jakirkham

 Genevieve Buckley
 GenevieveBuckley

 Marvin Albert
 m-albert

dask-image
2023.08.1+1.g86b56a7.d
documentation

Search the docs ...

Installation

Quickstart

Function Coverage

API

Contributing

Credits

History

Theme by the Executable Book
Project



Function Coverage

Coverage of dask-image vs scipy ndimage functions

This table shows which SciPy ndimage functions are supported by dask-image.

Function name	SciPy ndimage	dask-image	dask-image GPU support
<code>affine_transform</code>	✓	✓	✓
<code>binary_closing</code>	✓	✓	✓
<code>binary_dilation</code>	✓	✓	✓
<code>binary_erosion</code>	✓	✓	✓
<code>binary_fill_holes</code>	✓		
<code>binary_hit_or_miss</code>	✓		
<code>binary_opening</code>	✓	✓	✓
<code>binary_propagation</code>	✓		
<code>black_tophat</code>	✓		

Practicals

Preparation:

Option 1:

Use the environment created in the [course preparation](#) and install some further packages to it:

```
mamba activate devbio-napari-env  
mamba install -c conda-forge dask-image ipycytoscape
```

Option 2:

Create a new environment from scratch:

```
mamba create --name <dask_course> python=3.9 devbio-napari pyqt dask-image ipycytoscape -c c  
mamba activate dask_course
```

Practical parts:

1. Learning dask basics
2. Parallelizing array operations
3. Creating a virtual stack viewer