浙江大学爱丁堡大学联合学院
ZJU-UoE Institute

Dealing with large datasets - Part 1

Nicola Romanò - nicola.romano@ed.ac.uk

Learning objectives

At the end of this lecture you should be able to:

- Describe the issues related to dealing with large (imaging) datasets
- Describe possible strategies to avoid these problems
- Use HDF5 file format to speed up access to large datasets.

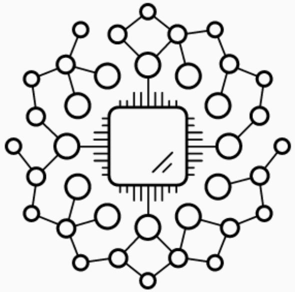
Learning objectives

At the end of this lecture you should be able to:

- Describe the issues related to dealing with large (imaging) datasets
- Describe possible strategies to avoid these problems
- Use HDF5 file format to speed up access to large datasets.

The next lecture will expand on this by introducing some tools specifically designed to work with large data!

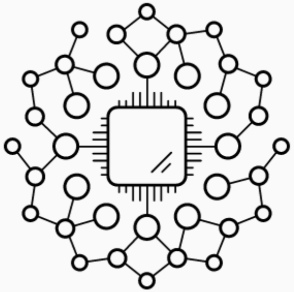
Big data



Oleksandr Panasovskyi, CC-BY 3.0

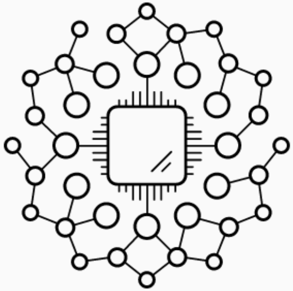
- Term popularised in the 1990s by John Mashey
- "Data sets with **sizes** beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed **time**."
  [Source: Wikipedia]

Big data


Oleksandr Panasovskyi, CC-BY 3.0

- Term popularised in the 1990s by John Mashey
- "Data sets with **sizes** beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed **time**."
[Source: Wikipedia]
- Recent advances in technology have allowed the generation of larger and larger amounts of data in many different fields, including biology.

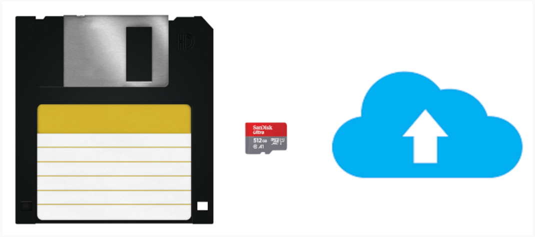
Big data


Oleksandr Panasovskyi, CC-BY 3.0

- Term popularised in the 1990s by John Mashey
- "Data sets with **sizes** beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed **time**."
  [Source: Wikipedia]
- Recent advances in technology have allowed the generation of larger and larger amounts of data in many different fields, including biology.
- Big data is commonly associated with the *5V* - Volume, Variety, Velocity, Value, Veracity (some sources also include Volatility).

The 5V of big data - Volume

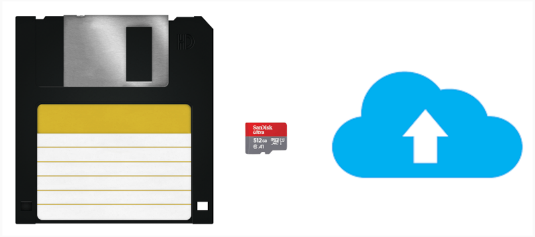
Volume refers to the increasing size of data that is being generated.



- Technical improvements (cameras, microscopes, staining techniques)
- Higher resolution images, longer time-lapses, larger 3D/4D stacks
- Automation allows for high-throughput imaging.
- Issues with storage availability (and associated costs).

The 5V of big data - Volume

Volume refers to the increasing size of data that is being generated.



- Technical improvements (cameras, microscopes, staining techniques)
- Higher resolution images, longer time-lapses, larger 3D/4D stacks
- Automation allows for high-throughput imaging.
- Issues with storage availability (and associated costs).

Ask yourself

- Where are the data going to reside?
- Are the data sensitive? Who is going to be able to access them?
- How long do you need to keep the data?

The 5V of big data - Variety

Variety refers to data coming from different sources.

- Data can be **structured** or **unstructured**
- Structured data is highly organized, standardized, searcheable
- Unstructured data has no predefined format, it is difficult to collect/analyse.

The 5V of big data - Variety

Variety refers to data coming from different sources.

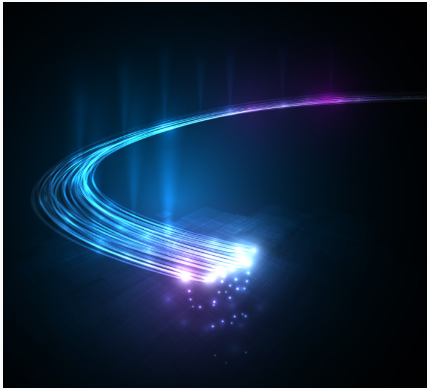
- Data can be **structured** or **unstructured**
- Structured data is highly organized, standardized, searcheable
- Unstructured data has no predefined format, it is difficult to collect/analyse.

Ask yourself

- How/where do I organize my data?
- Which file types/standards do I use?
- Will I be able to query the data easily in the future?

The 5*V* of big data - Velocity

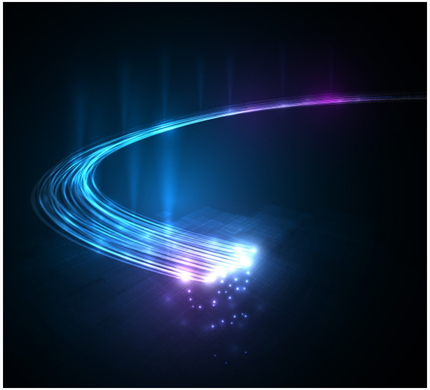
Velocity relates to the speed of data generation.



- Data are generated at ever-increasing speed...
- ... sometimes faster than our ability to generate insights from them.

The 5V of big data - Velocity

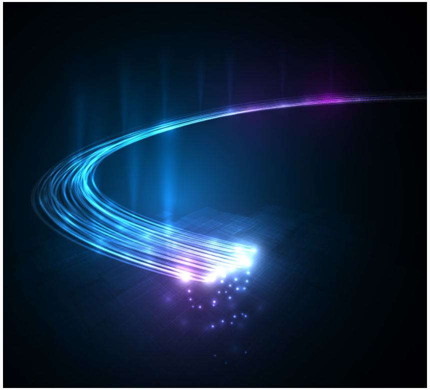
Velocity relates to the speed of data generation.



- Data are generated at ever-increasing speed...
- ... sometimes faster than our ability to generate insights from them.
- Real-time acquisition of images is often not matched by real-time analysis.

The 5V of big data - Velocity

Velocity relates to the speed of data generation.



- Data are generated at ever-increasing speed...
- ... sometimes faster than our ability to generate insights from them.
- Real-time acquisition of images is often not matched by real-time analysis.

Ask yourself:

- Can I automate data processing?
- Can I speed it up?
- Will I be able to interpret the results afterwards?
- Can my analysis be run in real-time?

The 5V of big data - Value

Value is concerned with the economical aspects of data.

Hans Splinter - Money! CC-BY-ND 2.0

- Globally, big data is expected to generate \$70 billion by 2024.
- Aside from monetary terms, value should also be considered in terms of improving healthcare and expanding human knowledge.

The 5V of big data - Value

Value is concerned with the economical aspects of data.

Hans Splinter - Money! CC-BY-ND 2.0

- Globally, big data is expected to generate $70 billion by 2024.
- Aside from monetary terms, value should also be considered in terms of improving healthcare and expanding human knowledge.

Ask yourself:

- What is the (economical) value of my data?
- How much does it cost to generate these data and analyse it?
- Will I be able to sustain this?

The 5V of big data - Veracity

Veracity refers to the accuracy of the data, and their biases.

Jared Cherup - Question mark block CC-BY-NC-ND 2.0

- Data coming from multiple sources may be discordant.
- Data generation may be unclear
- Methodology matters!

The 5V of big data - Veracity

Veracity refers to the accuracy of the data, and their biases.

Jared Cherup - Question mark block CC-BY-NC-ND 2.0

- Data coming from multiple sources may be discordant.
- Data generation may be unclear
- Methodology matters!

Ask yourself:

- Is my data source trustable?
- Is the methodology used to acquire those images allowing me to answer the question I need to answer?
- Can I (automatically) detect and possibly correct inaccuracies in the data?

Some practical examples

High-throughput screening

- You are performing high-throughput imaging of multiwell plates.
- You acquire 1024 x 1024 images of 384 well plates, taking 10 images/well with 50 z-planes/image and 3 colour channels.

Some practical examples

High-throughput screening

- You are performing high-throughput imaging of multiwell plates.
- You acquire 1024 x 1024 images of 384 well plates, taking 10 images/well with 50 z-planes/image and 3 colour channels.
- Total of 600k images, assuming 2Mb for one image $\rightarrow$ 10Tb of space/plate.

Some practical examples

High-throughput screening

- You are performing high-throughput imaging of multiwell plates.
- You acquire 1024 x 1024 images of 384 well plates, taking 10 images/well with 50 z-planes/image and 3 colour channels.
- Total of 600k images, assuming 2Mb for one image $\rightarrow$ 10Tb of space/plate.
- Assuming it takes 20 seconds to process each image, you will need 3000+ hours (a few months!) to process the whole plate.

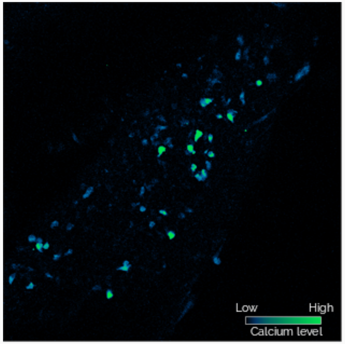
Some practical examples

High-throughput screening

- You are performing high-throughput imaging of multiwell plates.
- You acquire 1024 x 1024 images of 384 well plates, taking 10 images/well with 50 z-planes/image and 3 colour channels.
- Total of 600k images, assuming 2Mb for one image $\rightarrow$ 10Tb of space/plate.
- Assuming it takes 20 seconds to process each image, you will need 3000+ hours (a few months!) to process the whole plate.
- If this is automated you may get several plates each week!

Some practical examples

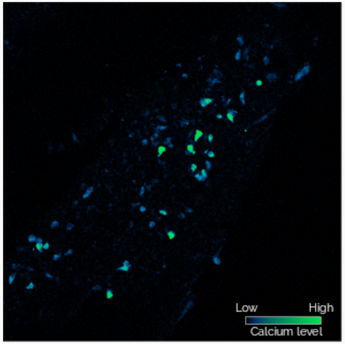
Real time analysis of cell activity



- You perform calcium imaging experiments on tissue slices
- You want to identify cells that drive the activity of other cells to optogenetically inhibit them

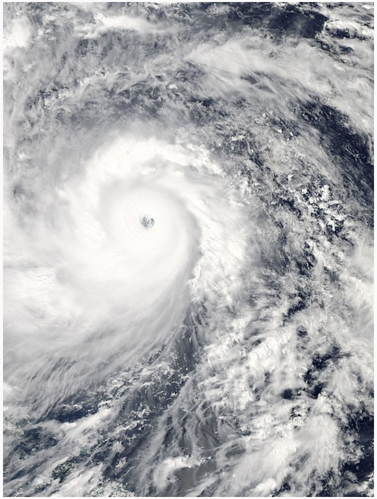
Some practical examples

Real time analysis of cell activity



- You perform calcium imaging experiments on tissue slices
- You want to identify cells that drive the activity of other cells to optogenetically inhibit them
- You need to feed your video to a script that automatically analyses it and controls the precise activation of a laser for optogenetic inhibition.

Danger, too much data ahead!

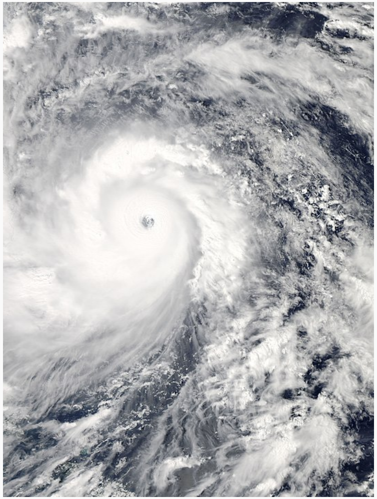


Souce: NASA, Public Domain

How not to drown in a sea of [image] data?

Problem-dependent solutions, today we will focus on:

Danger, too much data ahead!


Souce: NASA, Public Domain

How not to drown in a sea of [image] data?

Problem-dependent solutions, today we will focus on:

- Hardware solutions
- Choice of file format
- Parallelization
- Lazy evaluation/loading
- ...

Hardware solutions

Raimond Spekking - CC BY-SA 4.0

- Use better hardware (faster CPU, more RAM, etc.)! This solves the problem ...to a point!

Hardware solutions

Raimond Spekking - CC BY-SA 4.0

- Use better hardware (faster CPU, more RAM, etc.)! This solves the problem …to a point!
- Using multi-core processor allow for **parallelization**, that is, performing more than one operation at the same time.

Hardware solutions

Raimond Spekking - CC BY-SA 4.0

- Use better hardware (faster CPU, more RAM, etc.)! This solves the problem ...to a point!
- Using multi-core processor allow for **parallelization**, that is, performing more than one operation at the same time.
- High-end computers can use the Graphics Processing Unit (**GPU**) to perform fast operations. GPUs allow for GPGPU (General-Purpose computing on Graphics Processing Units), parallel computing between CPU and GPU, through specific languages such as CUDA or OpenCL.

Hardware solutions

Raimond Spekking - CC BY-SA 4.0

- Use better hardware (faster CPU, more RAM, etc.)! This solves the problem ...to a point!
- Using multi-core processor allow for **parallelization**, that is, performing more than one operation at the same time.
- High-end computers can use the Graphics Processing Unit (**GPU**) to perform fast operations. GPUs allow for GPGPU (General-Purpose computing on Graphics Processing Units), parallel computing between CPU and GPU, through specific languages such as CUDA or OpenCL.
- In 2016 Google also introduced Tensor Processing Units, circuits specifically developed to improve the speed of artificial intelligence and machine learning tasks.

[Image] file formats for big data

When it comes to large image (and not only) files, the HDF5 and Zarr file formats are the format of choice.

They allow for

- Chunked read/write operations of multidimensional data
- Hierarchical storage of multiple images/datasets
- Compression (lossless or lossy)
- Saving image with associated metadata.

HDF5 and Zarr

HDF (Hierarchical Data Format).

- Allows storing large heterogeneous data using a directory structure.
- 20+ years old, widely used format, accessible from many languages (Python, R, C/C++, Matlab, Java, ...).
- Can be used in Python through the *h5py* package (or *pytables*).
- The HDF5 specification is extremely long and complex

HDF5 and Zarr

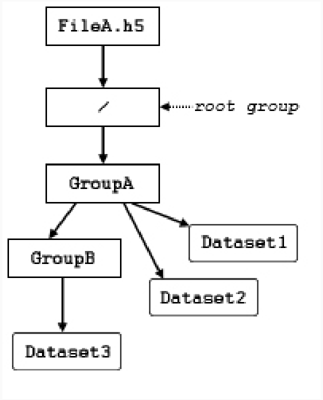
HDF (Hierarchical Data Format).

- Allows storing large heterogeneous data using a directory structure.
- 20+ years old, widely used format, accessible from many languages (Python, R, C/C++, Matlab, Java, ...).
- Can be used in Python through the *h5py* package (or *pytables*).
- The HDF5 specification is extremely long and complex

Zarr

- Relatively new project (released in 2015)
- Python-only project, use with the *zarr* package
- Built on top of Numpy.
- Similar hierarchical structure to HDF5
- Allows concurrent access to the files

Hierarchical storage (HDF5)



Example of hierarchical structure in a HDF5 file - Source: HDF5 documentation

- Storage of multiple datasets in a tree like manner
- All groups are stored under the base group called root or /.
-
- Circular and self-references possible (use with caution!)

A simple example - writing to HDF5

```
import numpy as np
import h5py
# Create some example data
example_data = np.random.randint(0, 255, size=(1024, 1024), dtype=np.uint8)
```

A simple example - writing to HDF5

```
import numpy as np
import h5py
# Create some example data
example_data = np.random.randint(0, 255, size=(1024, 1024), dtype=np.uint8)

# Open the file in write mode
f = h5py.File("test.h5", "w")
# Create a dataset called and save our data in it
f.create_dataset("Experiment1", data = example_data)
f.close()
```

A simple example - writing to HDF5

```
import numpy as np
import h5py
# Create some example data
example_data = np.random.randint(0, 255, size=(1024, 1024), dtype=np.uint8)

with h5py.File("test.h5", "w") as f:
    f.create_dataset("Experiment1", data = example_data)
```

Note: opening in "w" mode **will overwrite** the file if it already exists. If you want to add to an existing file open in "a" (append) mode.

A simple example - reading to HDF5

```
# Open the file in write mode
f = h5py.File("test.h5", "r")
print(f)
<HDF5 file "test.h5" (mode r)>
```

A simple example - reading to HDF5

```
# Open the file in write mode
f = h5py.File("test.h5", "r")
print(f)
<HDF5 file "test.h5" (mode r)>

print(f.keys())
<KeysViewHDF5 ['Experiment1']>

print(f['Experiment1'])
<HDF5 dataset "Experiment1":  shape (1024, 1024), type "|u1">

# Retrieve data into a numpy vector
img = f['Experiment1'][:]
```

Compression

We can compress HDF5 files to use less space on disk

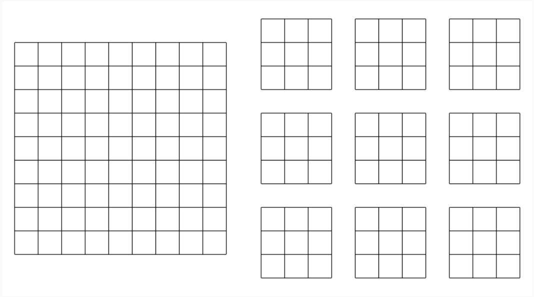
```
from skimage.io import imread
import os
img = imread("largeimg.tif")

with h5py.File("large.h5", "w") as f:
   f.create_dataset("uncompressed", data=img)
with h5py.File("compressed.h5", "w") as f:
   f.create_dataset("compressed", data=img, compression="gzip",
   compression_opts=5)
print(os.path.getsize("large.h5"))
57411584
print(os.path.getsize("compressed.h5"))
18824931
```

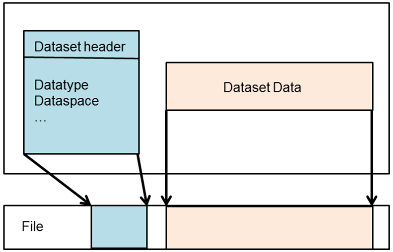
Amount of compression varies depending on the image content!

Chunking

HDF5 can save data in chunks. This allows for fast retrieval of parts of large arrays.

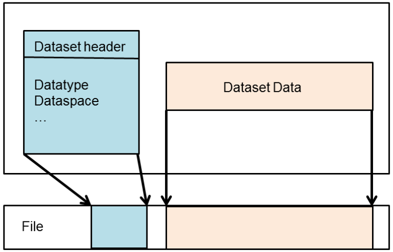


Contiguous vs Chunked layouts

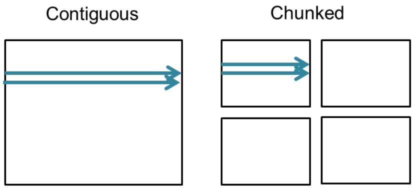


Contiguous Storage Layout

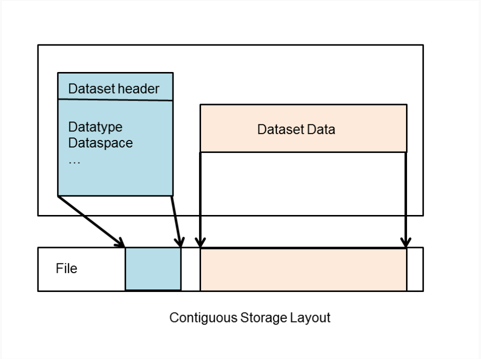
Contiguous vs Chunked layouts

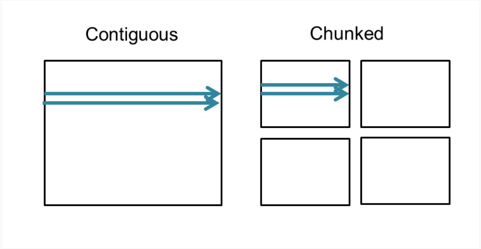


Contiguous Storage Layout



Contiguous vs Chunked layouts



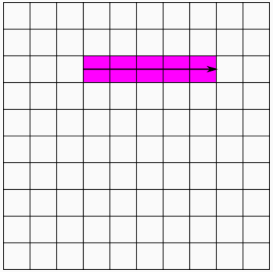


Each chunk can be accessed (for reading and writing) independently. They are stored independently in the file. If the file is compressed, each chunk will be compressed independently. Chunks can be of any shape.

Accessing part of an HDF5 file

What happens when we try to access part of a file?

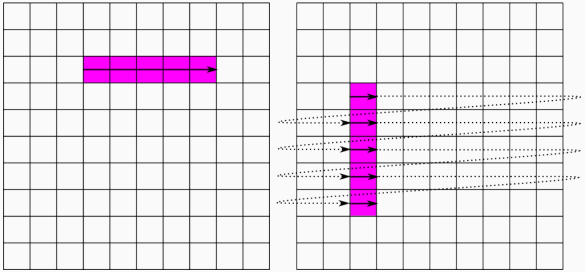
```
f['Experiment1'][3, 4:9])
```



Accessing part of an HDF5 file

What happens when we try to access part of a file?

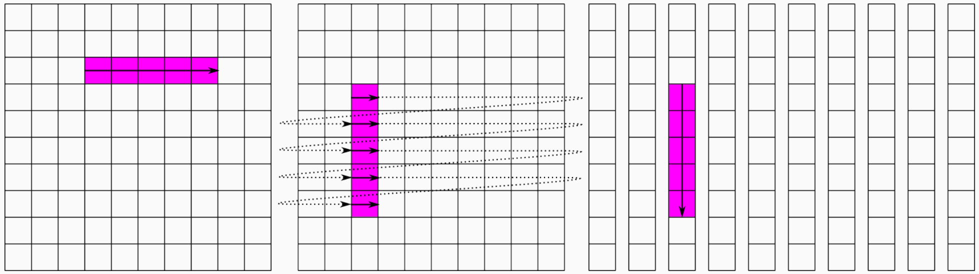
```
f['Experiment1'][3, 4:9])
```



Accessing part of an HDF5 file

What happens when we try to access part of a file?

```
f['Experiment1'][3, 4:9])
```



Creating a chunked HDF5 file

```
test = np.ones((1000, 1000, 5))
with h5py.File("chunked.h5", "w") as f:
   f.create_dataset("chunked", data=test, chunks=(100, 100, 5))
```

Creating a chunked HDF5 file

```
test = np.ones((1000, 1000, 5))
with h5py.File("chunked.h5", "w") as f:
   f.create_dataset("chunked", data=test, chunks=(100, 100, 5))
```

Important:

- Chunk size must be smaller than data size
- A wisely chosen chunk size will speed up your processing, a badly chosen chunk size will actually slow you down!
- Don’t make chunks too big. If chunks don’t fit in memory, you won’t be able to read them either!
- Don’t make chunks too small. There is some overhead in reading/writing chunks, and too many read/write operations may be costly.

Coming up...

In the next lecture we will see how to use Python for parallel processing and we will learn to use the *dask* library for optimizing computation on large datasets.