Introduction to HDF5

Mark Cheeseman
Mark.Cheeseman@csiro.au

ASTRONOMY DATA AND COMPUTING SERVICES

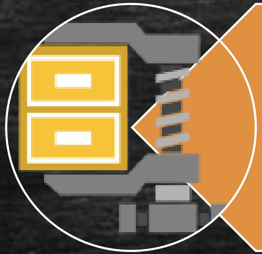
What we will cover

- Introduction to HDF5
- Chunking
- Compression
- Parallel I/O in HDF5 (time permitting)

HDF The HDF Group

What is HDF5?

Components of HDF5

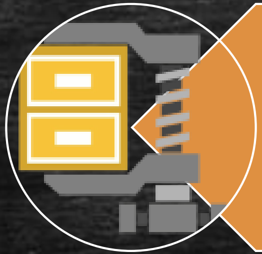


Low level file format that depicts how data is stored

Logical model that describes how model is organized

A fully featured software package with libraries, APIs and tools

Components of HDF5



Low level file format that depicts how data is stored

Logical model that describes how model is organized

A fully featured software package with libraries, APIs and tools

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Quick Description

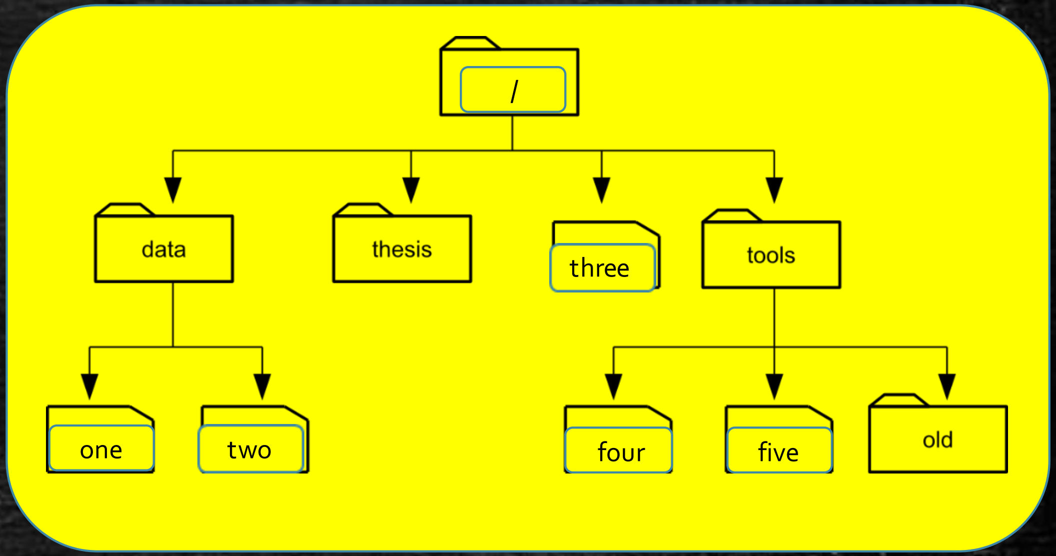
HDF5 is a binary data format that:

1. is self-describing
2. is machine independent
3. has wide support with industry and application developers
4. contains features targeting usability and performance
5. comes with a set of utilities
6. has multiple languages supported (C/C++, FORTRAN, Java, Python, R, MATLAB, …)

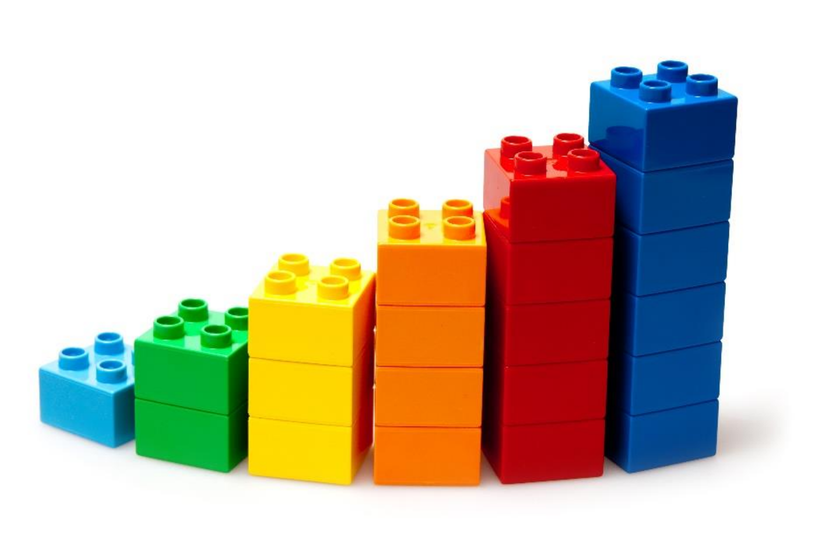
ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Data Organization in a HDF5 File

- Contents resemble a filesystem with:
  - subdirectories (called groups)
  - files (called datasets)
- All components have associated metadata
- Datasets have their own organization:
  - datatype
  - dataspace



File Basics



HDF5 File Walkthrough

```
HDF5 "myfile.h5" {
GROUP "/" {
  DATASET "MyData" {
    DATATYPE  H5T_STD_I64LE
    DATASPACE  SIMPLE { ( 100 ) / ( 100 ) }
    ATTRIBUTE "Description" {
      DATATYPE  H5T_STRING {
        STRSIZE H5T_VARIABLE;
        STRPAD H5T_STR_NULLTERM;
        CSET H5T_CSET_ASCII;
        CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
    }
  }
}
}
```

ASTRONOMY DATA AND COMPUTING SERVICES

HDF5 File Walkthrough

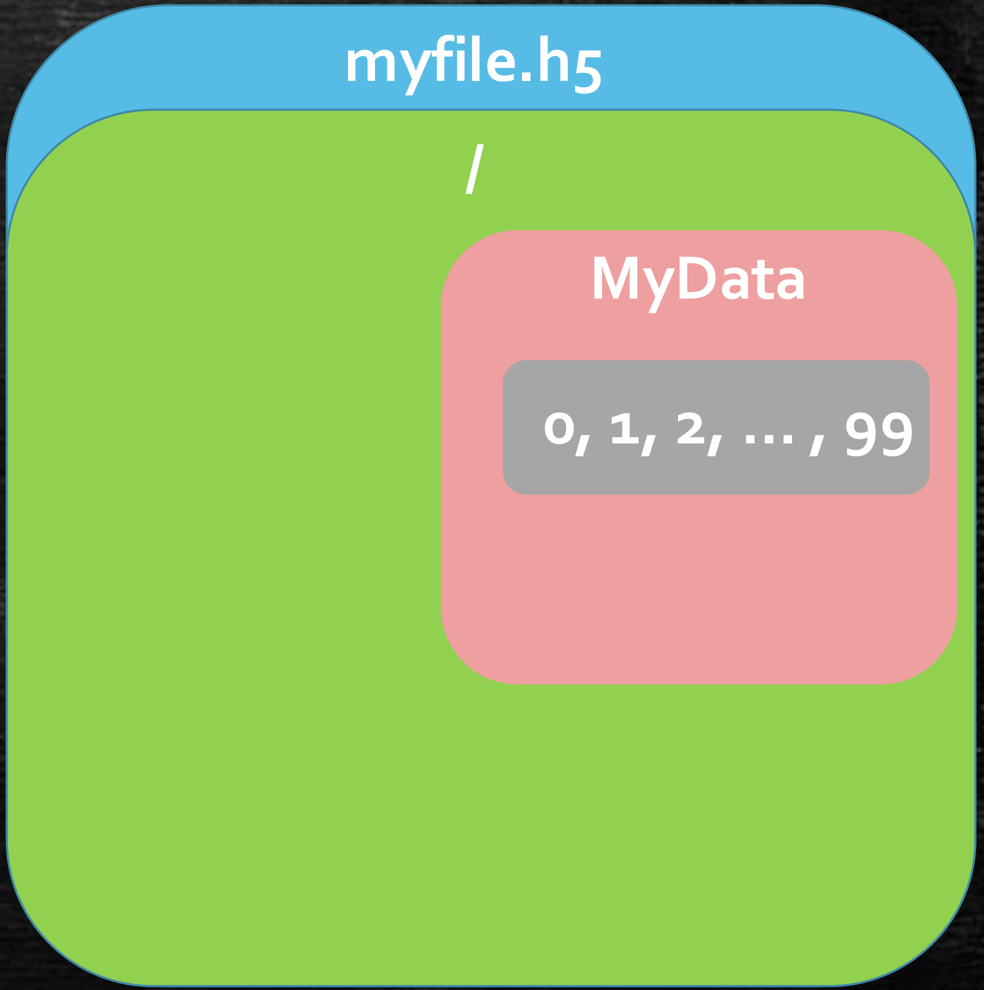
```
HDF5 "myfile.h5" {
GROUP "/" {
  DATASET "MyData" {
    DATATYPE  H5T_STD_I64LE
    DATASPACE  SIMPLE { ( 100 ) / ( 100 ) }
    ATTRIBUTE "Description" {
      DATATYPE  H5T_STRING {
        STRSIZE H5T_VARIABLE;
        STRPAD H5T_STR_NULLTERM;
        CSET H5T_CSET_ASCII;
        CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
    }
  }
}
}
```

Groups

- Can be thought of as directories or containers for HDF5 objects
- Used for imposing structure
  - Will have a root ("/") group by default
  - Can think of the File object being the same as the Root group object
- Every group will have two components
  a) Table listing all objects it contains
  b) Header containing its name
     - additional attributes can be added by user

HDF5 File Walkthrough

```
HDF5 "myfile.h5" {
GROUP "/" {
  DATASET "MyData" {
    DATATYPE H5T_STD_I64LE
    DATASPACE SIMPLE { ( 100 ) / ( 100 ) }
    ATTRIBUTE "Description" {
      DATATYPE H5T_STRING {
        STRSIZE H5T_VARIABLE;
        STRPAD H5T_STR_NULLTERM;
        CSET H5T_CSET_ASCII;
        CTYPE H5T_C_S1;
      }
      DATASPACE SCALAR
    }
  }
}
}
```



Datasets

- Consist of : a) Header and b) Data array
- Headers include the following required metadata
  - **Name** of the dataset
  - **Datatype**. Four categories of datatypes are supported
    - atomic (int, float, etc),
    - native (system-specific atomics)
    - compound (eg. structure)
    - named (when user explicitly renames another datatype)
  - **Dataspace** / dimensionality of the data array
  - **Storage Layout.** How data array is physically stored
    - Can be *contiguous*, *chunked* or *compact*

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Dataspaces & Datasets

- A dataspace describes the dimensionality of the data array
- Can have fixed, flexible or unlimited dimensions
  - The current and maximum length of each dimension is displayed
- Commonly used dataspaces descriptors
  - **Scalar**: contains only 1 element of any datatype
  - **Simple**: multi-dimensional array. Fixed or variable dimensions.
  - **Null**. Empty set
- Can cover all or part of a dataset
  - Partial I/O operations require dataspaces smaller than dataset

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Dataset Layout

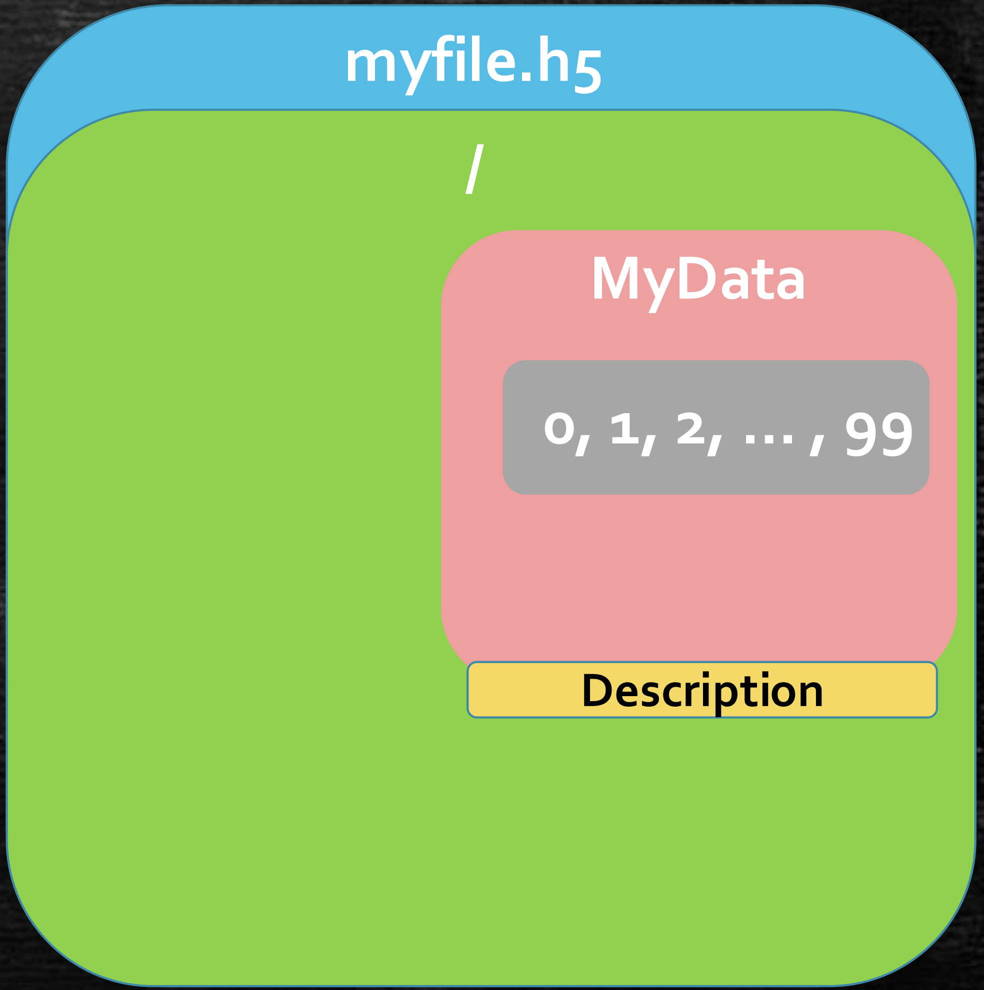
Data arrays <u>will</u> be:

- contiguous
- Immutable (consistent datatype throughout)
- Have different storage options
  - chunked
  - compressed

ADACS ASTRONOMY DATA AND COMPUTING SERVICES

HDF5 File Walkthrough

```
HDF5 "myfile.h5" {
GROUP "/" {
  DATASET "MyData" {
    DATATYPE  H5T_STD_I64LE
    DATASPACE  SIMPLE { ( 100 ) / ( 100 ) }
    ATTRIBUTE "Description" {
      DATATYPE  H5T_STRING {
        STRSIZE H5T_VARIABLE;
        STRPAD H5T_STR_NULLTERM;
        CSET H5T_CSET_ASCII;
        CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
    }
  }
}
}
```



Attributes

- Important tool is making your data readable and future-proof. Could include:
  - Title / description of dataset
  - Generation date
  - Units of measurement
- Can be attached to:
  - Groups
  - Datasets
  - Named datatypes
- Composed of two components:
  - **Name**
  - **Value**

h5py

- Python-based package that allows access to HDF5
  - Most features available
- Easy to use, high-level syntax
  - Commands that take multiple lines in C/Fortran can be done in 1 line
  - Low level API access to more complex/advanced features
- Not officially supported by HDF Group
  - Examples and info given on the HDF5 website

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

File Creation Demo

- In this demo, we do the following:
  - Create a new HDF5 file
  - Create a dataset and read/write data to/from it
  - Add attributes to a dataset
- To do:
  - Add a second dataset containing an array of 100 integers
  - Add an "units" attribute to both datasets. Values of units can be anything.

Demo: How to Run

1. Click on the terminal in your JupyterHub screen
2. Connect via ssh to the Athena cluster located at Pawsey

   ssh couXXX@athena.pawsey.org.au
3. Go into the HDF5 demo directory

   cd HPC-Workshop/HDF5/file_creation
4. Submit the demo jobscript to SLURM

   sbatch jobscript.slurm
5. Look at the output with the vi text editor

   vi HDF5_Test
6. Exit vi by typing :q and pressing ENTER

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Chunking

Memory Storage in HDF5 Datasets

- Default memory layout is a contiguous ordering of elements.
  - "linearizing" of multi-dimensional arrays
  - differs depending on underlying API used
    - C/Python uses row-wise ordering
    - FORTRAN uses column-wise ordering
  - perfectly fine if all accesses will be fully contiguous as well

A1	A2	A3
A4	A5	A6
A7	A8	A9

A1	A2	A3	A4	A5	A6	A7	A8	A9

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Memory Storage in HDF5 Datasets

Non-contiguous memory accesses lead to performance issues

- Ex: reading 1st column of a 2D matrix stored row-wise contiguously

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13

At least 3 (if not all 4) rows will be read in. This leads to cache swapping.

Chunking

Chunking is where the user changes the storage order to accommodate frequently ***expected*** access patterns to the dataset.

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

Contiguous layout

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

4-way Chunking

ASTRONOMY DATA AND COMPUTING SERVICES

Chunking

Read/write access now involves the accessing of appropriate chunks.

Consider the example of reading only the 1st column of values:

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

Only 2 chunks (grey and green) need to be accessed. Only 8 instead of 13 elements are read.

Imagine if global dimensions are huge! Number of transactions saved would be substantial.

Proper chunk sizing for your access pattern will yield BIG performance gains

ASTRONOMY DATA AND COMPUTING SERVICES

Chunking

Read/write access now involves the accessing of appropriate chunks.

Consider the example of reading only the 1st column of values:

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

What would be a better chunk pattern to use?

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Chunking

Read/write access now involves the accessing of appropriate chunks.

Consider the example of reading only the 1st column of values:

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

Column-wise chunking would be the most efficient

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Additional Benefits of Chunking

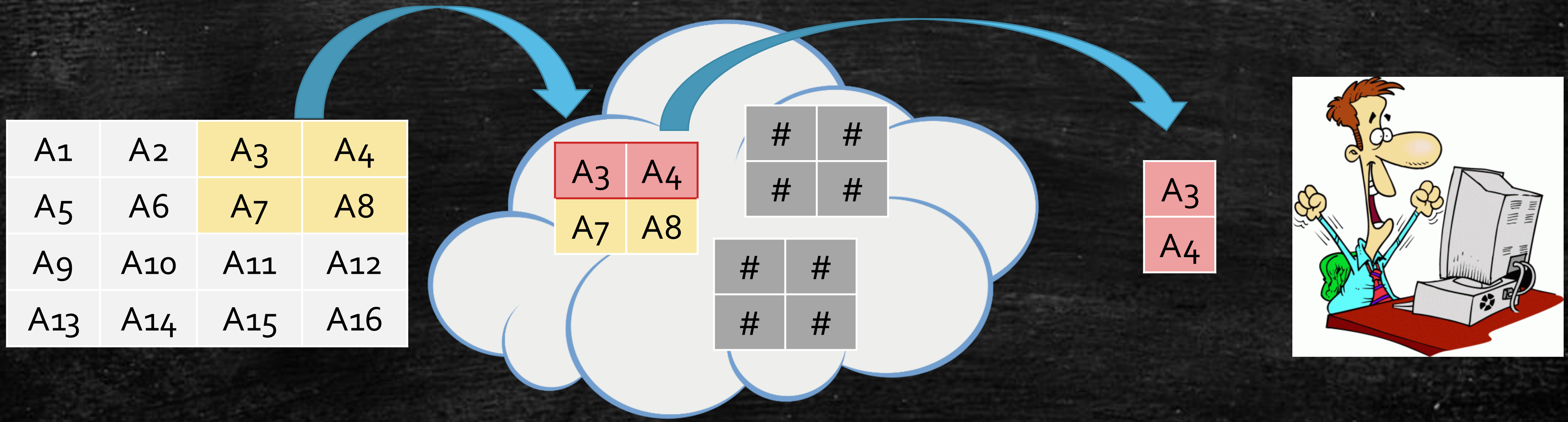
- Enables easy and efficient dataset re-sizing
- Allows the use of compression
- Allows the use of parallel I/O
- Has its own caching mechanism

Additional Benefits of Chunking

- Enables easy and efficient dataset re-sizing
- Allows the use of compression
- Allows the use of parallel I/O
- Has its own caching mechanism

~~Advanced~~ Chunk Caching

- During a read, chunks are read into a cache before finally landing in the user buffer.



Advanced Chunk Caching

All subsequent reads and writes to that chunk will be in memory (no disk access) until the chunk is evicted from the cache

- Memory limit of the cache is reached
- Max number of trackable chunks in cache reached

Size of the cache and the tracking table (hash table) are user-controlled

- Default cache size is 1MB

Dramatic performance gains are possible using optimal chunk and cache sizes

ADACS ASTRONOMY DATA AND COMPUTING SERVICES

Gotchas

- There are limits to chunking in a dataset
  - Cannot have more than 4,294,967,295 elements in a chunk
  - A chunk cannot exceed 4GB
  - Chunk dimensions cannot exceed dataset dimensions
- Be careful when setting chunk dimensions
  - **Too small**: latency issues arise. There's overhead in creating and tracking chunks.
  - **Too large**: lose the performance benefit. Caching effects are negated.

Common Issues with Chunk Caching

1. chunk cache not large enough
   - It needs to be large enough to avoid unnecessary swapping to/from disk
2. hash table is constantly overfilled and refreshed
   - Too many chunks are in play and can't all be tracked
   - Leads to the chunk cache only being partially filled
   - happens if using a chunk size that is too small
3. It may be more efficient to NOT to use chunking if:
   - Cannot allocate enough memory to the cache to be efficient
   - Using a contiguous memory access pattern

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Chunking Demo

- At the end of demo, we should be able to do the following:
  - Create a HDF5 dataset with chunking enabled
  - Inquire if a dataset has chunking and, if so, what are the chunk dimensions
  - Get and modify the size of a chunk cache
- To do:
  - Observe any read performance difference between chunked and non-chunked datasets with the same dimensions
    - Contiguous reads
    - Non-contiguous reads
  - Investigate the performance improvement with modifying chunk cache size

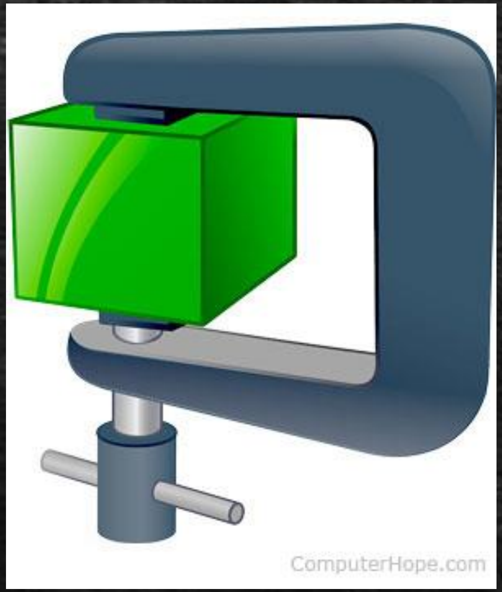
ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Demo: How to Run

1. Click on the terminal in your JupyterHub screen
2. Connect via ssh to the Athena cluster located at Pawsey
   ssh couXXX@athena.pawsey.org.au
3. Go into the HDF5 demo directory
   cd HPC-Workshop/HDF5/chunking
4. Submit the demo jobscript to SLURM
   sbatch jobscript.slurm
5. Look at the output with a text editor (vi, emacs)
   vi HDF5_Test
6. Exit vi by typing :q and pressing ENTER

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

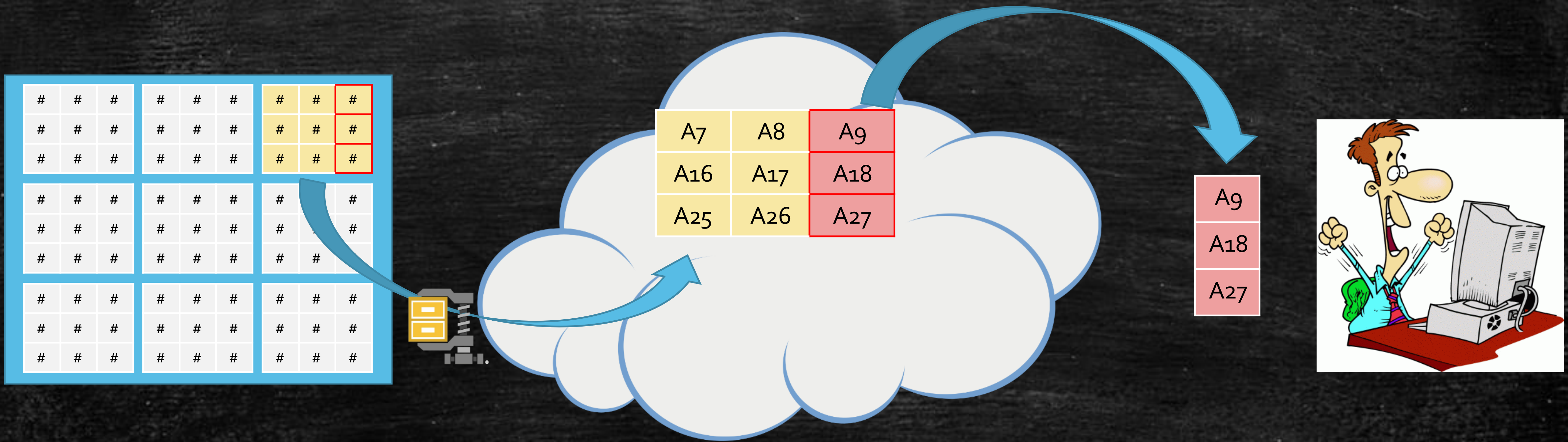
Compression



Compression in HDF5

- Compression is considered a *filtering* operation
  - Supported compression libraries are referred to as *filters*
- HDF5 has native support for GNU zip and Szip
  - h5py also natively supports LZF
- Compression can only be done on a chunked dataset
  - Individual chunks are compressed
- Compression cannot be used with parallel writes
  - Reads are ok.

Reading a Compressed Dataset



Common Compression Gripes

- **My file size didn't decrease [much]**
  - HDF5 wasn't configured properly with the desired filter
  - Some compression libraries work better on different data types
    - GNU zip works better on floating point data than Szip
    - Szip won't work on compound datatypes
- **Compression operation takes too long**
  - higher compression levels doesn't guarantee a smaller file size
    - Depends on filter and datatypes
  - GNU zip levels 6 and 9 achieve comparable compression ratios but level 9 requires much more time to compress/decompress data

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Improving Read Performance

- Optimal chunk size selection
  - Make sure it covers the expected access pattern to the dataset
  - Large enough to limit memory-disk swapping
- make chunk cache size sufficiently large enough
  - Remember that compression/decompression only occurs when a chunk leaves/enters the cache
- Change access pattern to coincide with chunking pattern
  - Alternative to re-sizing the chunk dimensions

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Compression Demo

- At the end of demo, we should be able to do the following:
  - Create a HDF5 dataset with GZIP compression enabled
  - Inquire if a dataset is compressed. If so, which filter was used
- To do:
  - Observe any read performance difference between compressed and non-compressed datasets with the same dimensions
    - Contiguous reads
    - Non-contiguous reads
  - Investigate the performance improvement with modifying chunk cache size

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

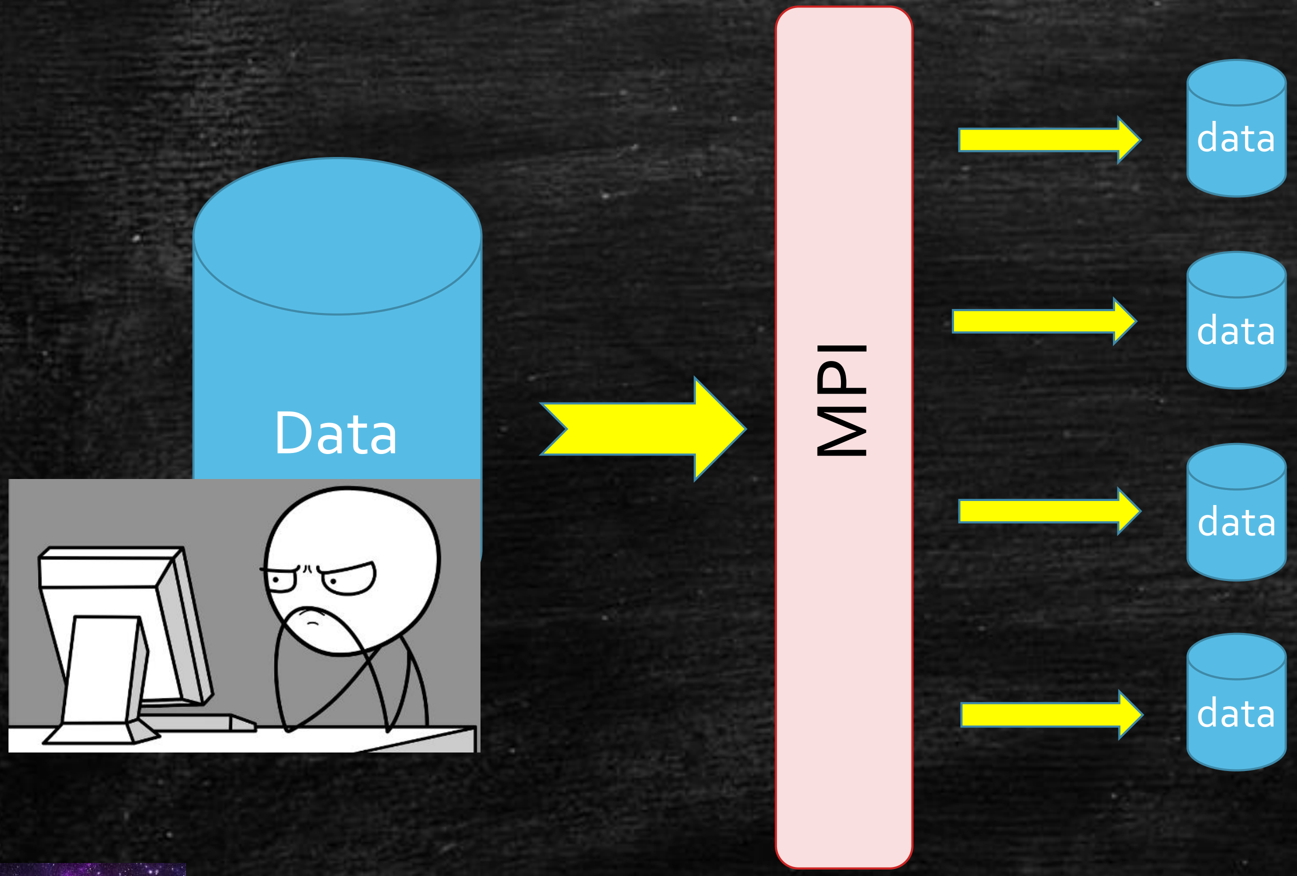
Demo: How to Run

1. Click on the terminal in your JupyterHub screen
2. Connect via ssh to the Athena cluster located at Pawsey
   `ssh couXXX@athena.pawsey.org.au`
3. Go into the HDF5 demo directory
   `cd HPC-Workshop/HDF5/compression`
4. Submit the demo jobscript to SLURM
   `sbatch jobscript.slurm`
5. Look at the output with a text editor (vi, emacs)
   `vi HDF5_Test`
6. Exit vi by typing `:q` and pressing ENTER

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

Parallel HDF5

Overview



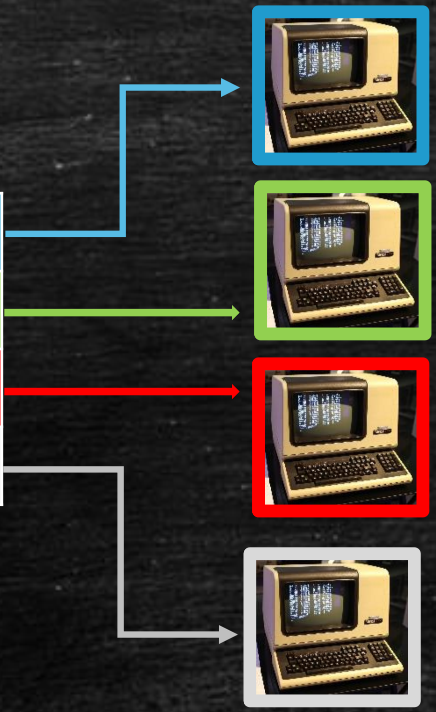
- Goal is to employ data parallelism for reads / writes
- Assumptions are:
  - using MPI in your code
  - running on a parallel filesystem
  - large datasets

Collective I/O

- Operations where all MPI tasks are involved simultaneously
  - All metadata operations are collective
- Best for large, uniformly spaced datasets
  - Goal is maximum bandwidth

Reading Rows in Row-Wise Storage

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16



- 2D matrix stored row-wise contiguously
- Have 4 tasks wanting to access different rows
- No overlay so reads/writes can be done concurrently
  - No state issues

Independent I/O

- MPI tasks can read/write independently any section of the open file
  - If sections don't overlap, writes can be done concurrently
- Why do this?
  - MPI layer can provide caching (useful for writing)
  - Concurrent I/O can achieve higher bandwidth than a serial I/O operation
- When should I use it?
  - Reading/writing data that is non-uniform (eg. Not contiguous or uniformly spaced)

Reading Columns in Row-Wise Storage

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

- matrix stored row-wise contiguously
- Have 2 tasks accessing data:
  - Red wants column 1
  - Green wants column 3
- No overlay so reads/writes can be done concurrently
- Just need to direct each task to *see* the column layout

Reading Columns in Row-Wise Storage

Couldn't these column accesses be done collectively?

YES, if

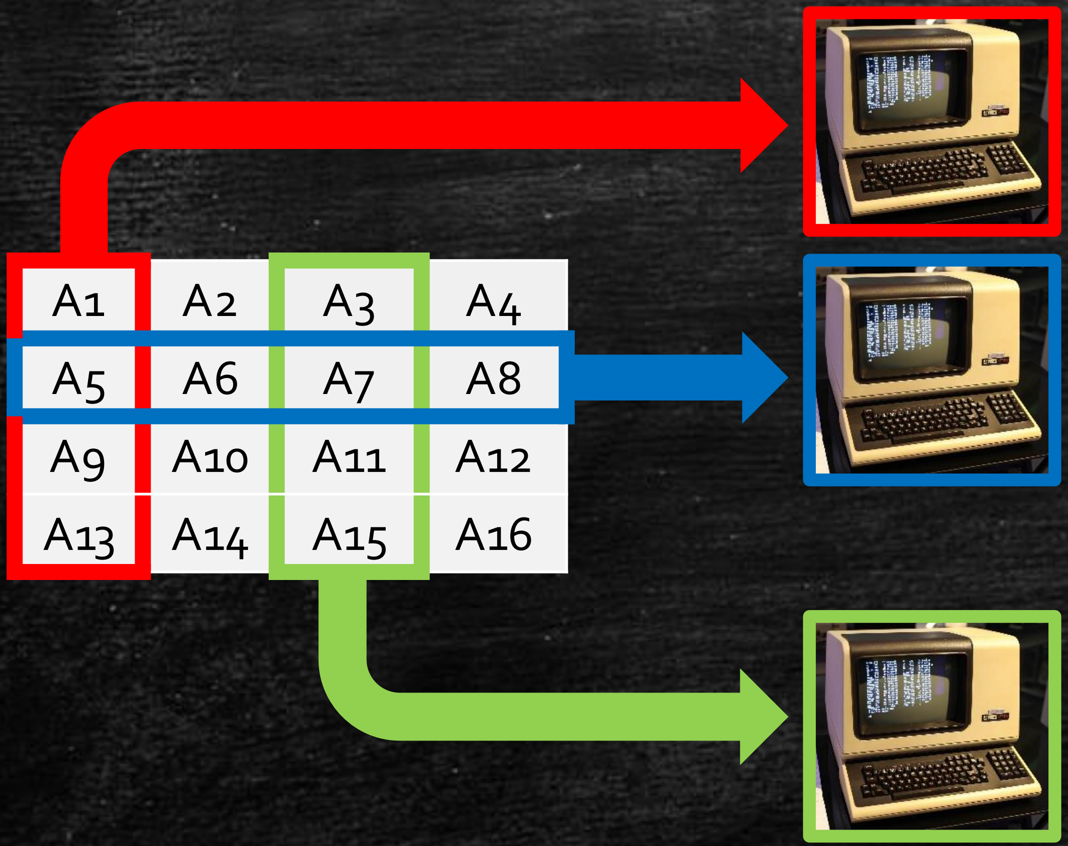
- all tasks are participating in the read/write operations
    - *can set 1 or more tasks to write no data*
- all accesses are non-lapping
- columns are uniformly separated

SO… why aren't we?

- 1 of the 3 above criteria is not met
- Collective operation may create a barrier to the parallel flow of the program

ASTRONOMY DATA AND COMPUTING SERVICES

Reading Columns in Row-Wise Storage



- Have 3 tasks accessing data:
  - Red wants column 1
  - Green wants column 3
  - Blue wants row 2
- Order of operations must be considered

Chunking and Parallel I/O

Pay attention to chunk dimensions and the block size of your parallel filesystem

- If there's no alignment, performance will suffer

Very, very, very important to know your data access pattern in your code

- Then set or find out your parallel file system strip size
- Then set an appropriate chunk size
- Then adjust chunk cache (if necessary)

ASTRONOMY DATA AND COMPUTING SERVICES

Parallel I/O and Compression

- Cannot perform parallel writes with compression enabled
  - Cannot predict size of compressed chunks
- Parallel reads can be performed