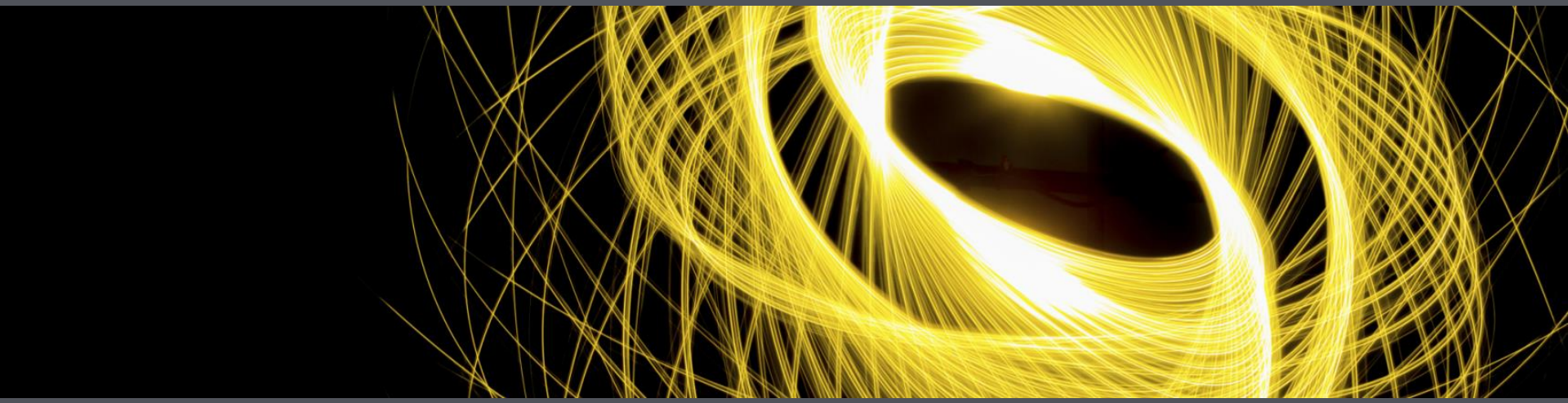
University of Reading

INVESTIGATING READ PERFORMANCE OF PYTHON AND NETCDF WHEN USING HPC PARALLEL FILESYSTEMS



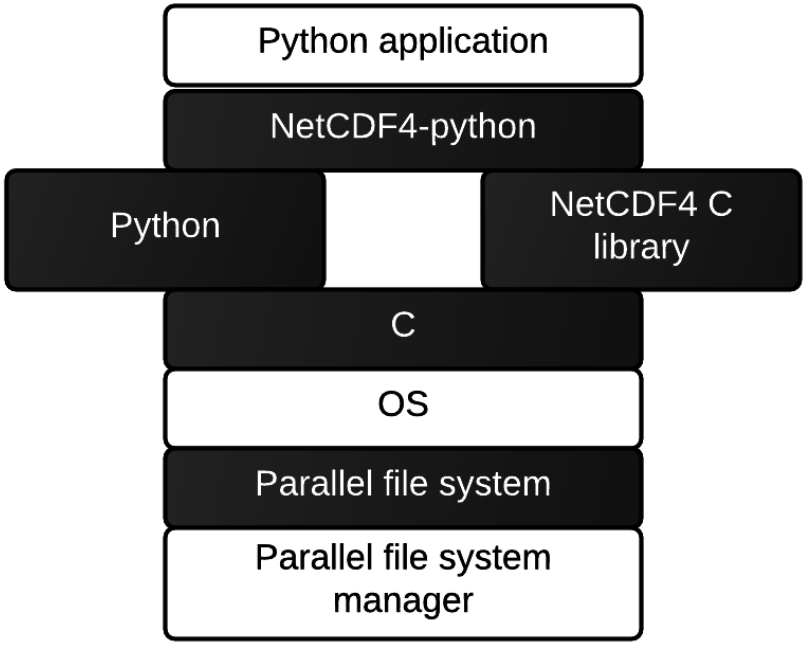
Matthew Jones | Jon Blower | Bryan Lawrence | Annette Osprey

m.jones3@pgr.reading.ac.uk

University of Reading

MOTIVATION

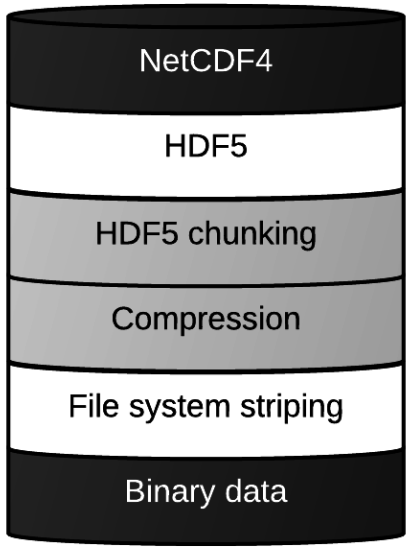
- Atmospheric data analysis getting difficult due to very large data sets (10-20PB for CMIP6)
- Parallel analysis one way of dealing with this
- But throwaway scripts
- Many different levels in software hardware stack can affect analysis script
- Need to understand contributions of each layer to write efficient analysis script



University of Reading

BACKGROUND

- NetCDF4 widely used to store data in Atmospheric science
- Binary format which is metadata rich
- Portable and can be used to store large datasets
- Built on HDF5, which allows chunking and compression
- File can also be striped in parallel file system
- All these can interact but initially need to see effect of NetCDF4

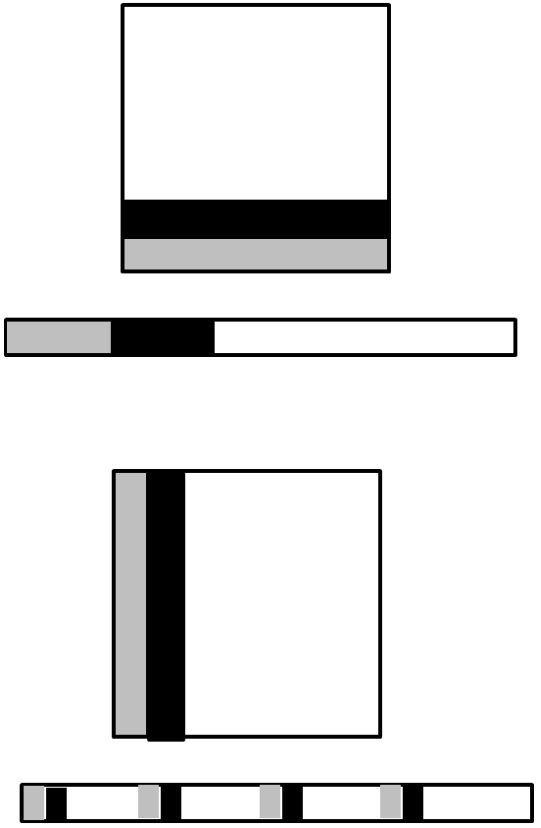


University of Reading

BACKGROUND

- As well as file type access pattern can play a large role
- In standard [t,z,y,x] layout only maps are contiguous
- For any other slice access pattern is striding or hopping through file

- Data generally stored in a repository so written once and read from many times



University of Reading

AIMS

- How much do read sizes affect read rate?
- How much does read pattern affect read rate?
- How much does using Python affect read rate?
- How much does using NetCDF4 files affect read rate?

University of Reading

METHOD

- Range of buffer sizes from 512b doubling to 1GiB
- Large files of ‘plain’ binary and NetCDF4 – double size of RAM
- 1D
- Build layers of complication:
  - C reading from plain binary file
  - Python reading from plain binary file
  - C reading from NetCDF4 file
  - Python reading from NetCDF4 file
- Three different read patterns:
  - Sequential
  - Striding
  - Random

University of Reading

METHOD

```
# Sequential read
f = open(filename )
for number of buffers :
    data = f.read(buffersize)

# striding read
f = open(filename)
readoffset = 0
for number of buffers :
    f.seek(readoffset)
    data = f.read(buffersize)
    readoffset = readoffset + 4*buffersize
```

University of Reading

METHOD

```
# Random read
f = open(filename)
readoffsets = genrandoffsets(length=100)
for offset in readoffsets :
    f.seek(offset)
    data = f.read(buffersize)
```

University of Reading

METHOD

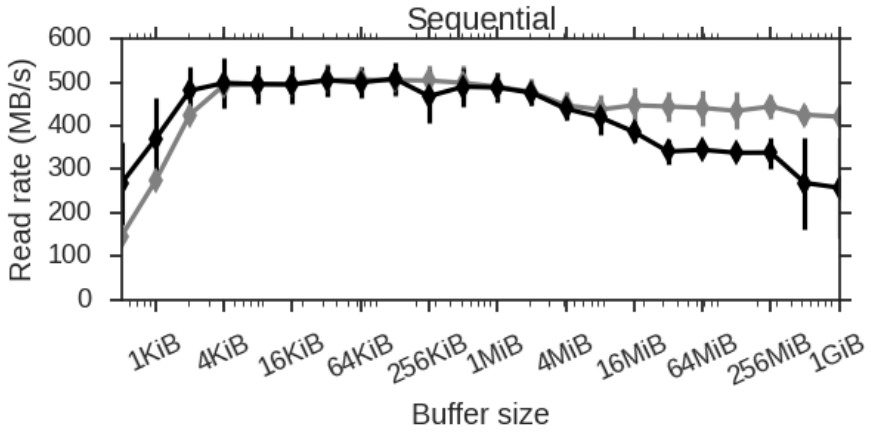
- Sequential read simulates best case scenario for a read
- Striding read simulates typical read pattern from slice which non sequential with regular strides
- Random read simulates worst possible read

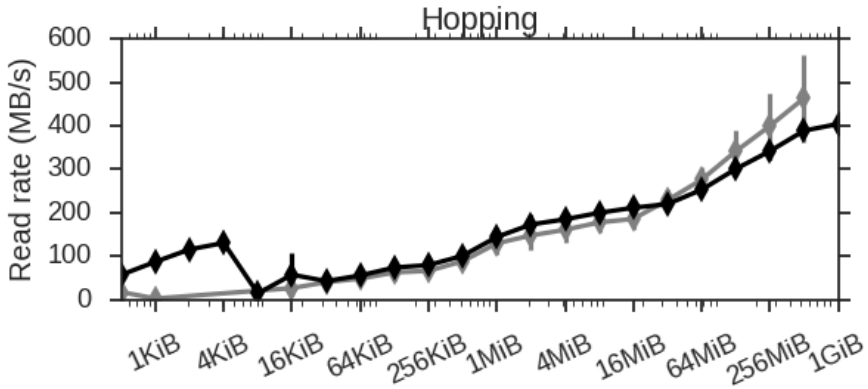
- Lustre and GPFS platforms only have C tests reading from a binary file to ascertain any major differences
- Panasas platform has all tests

University of Reading

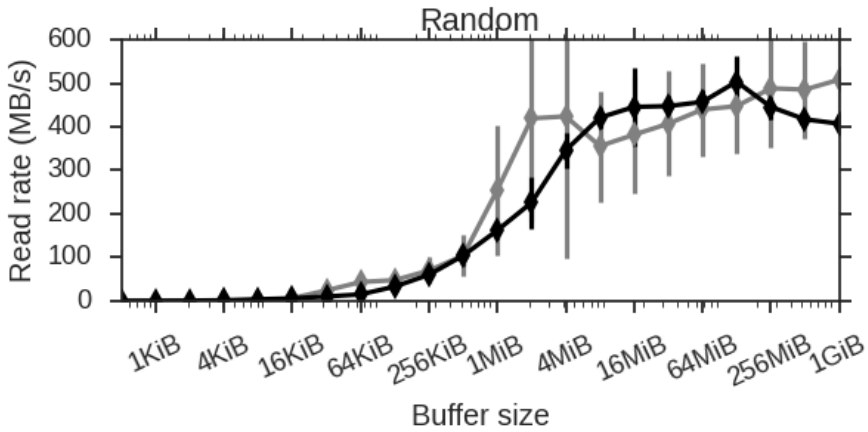
BASELINE PERFORMANCE

- Reads using python and C from 'plain' binary file on Panasas





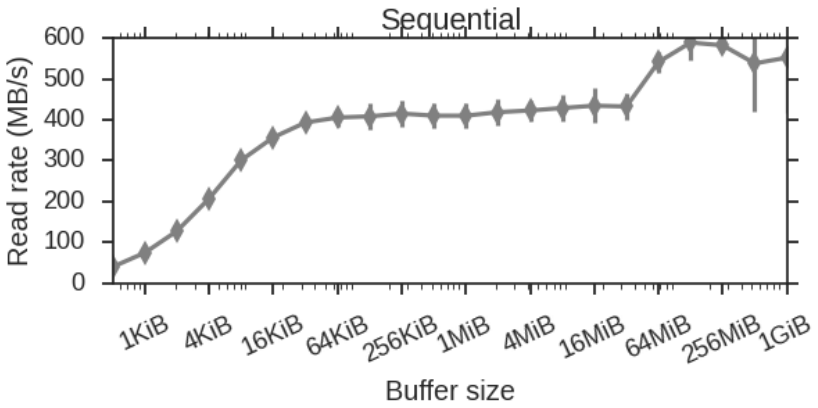
Grey – C
Black - python

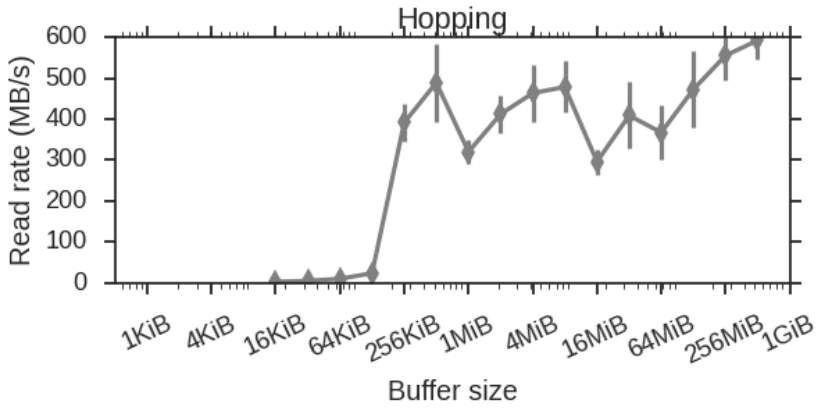


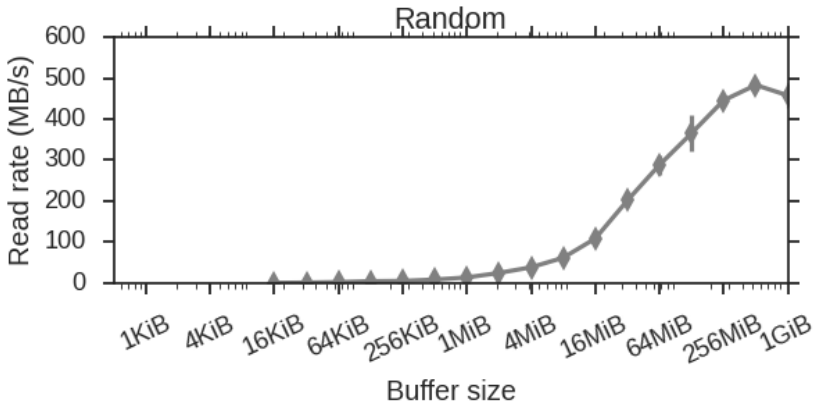
University of Reading

BASELINE PERFORMANCE

- Reads using C from 'plain' binary file on Lustre



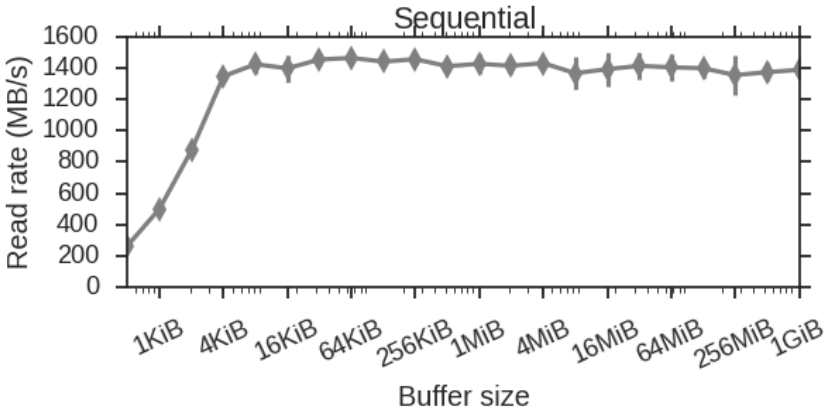


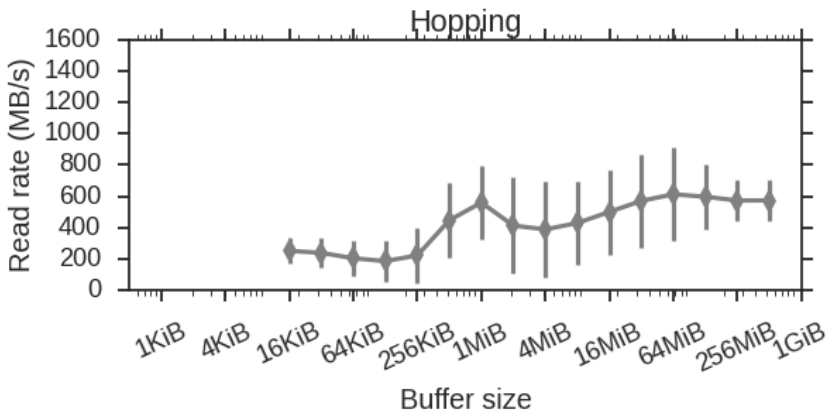


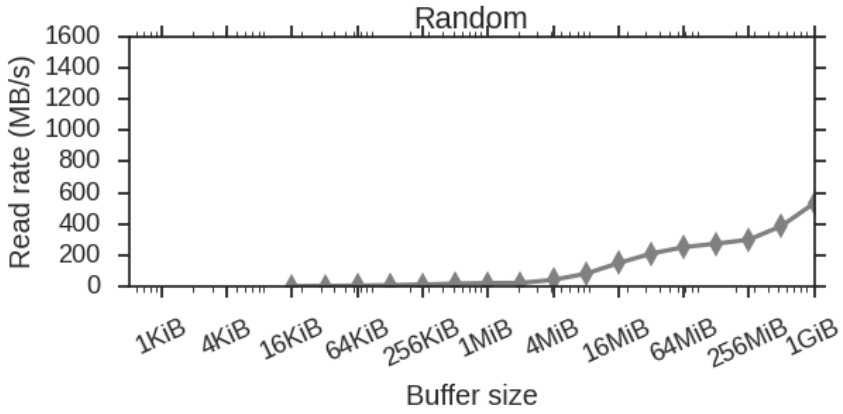
University of Reading

BASELINE PERFORMANCE

- Reads using C from 'plain' binary file on GPFS







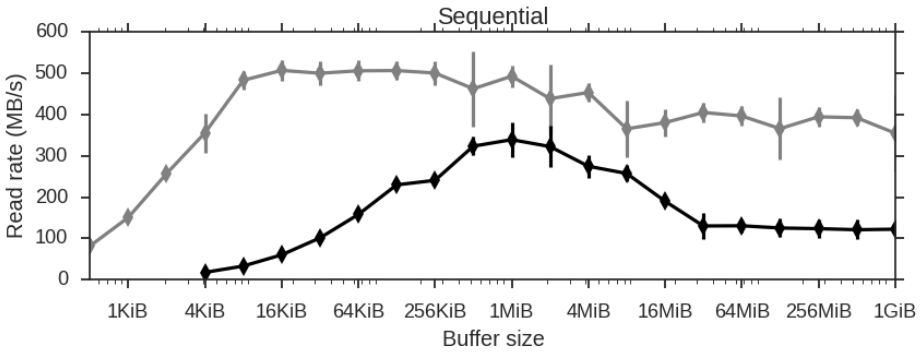
University of Reading

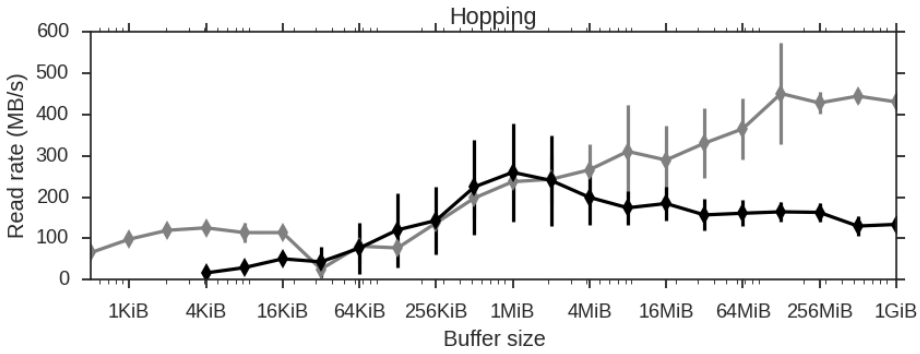
BASELINE PERFORMANCE

- Little difference between C and python when reading from plain binary files
- Different read patterns have large effect on read rate – due to read ahead on sequential reads
- Drop of read rate at small buffer size agrees with other literature

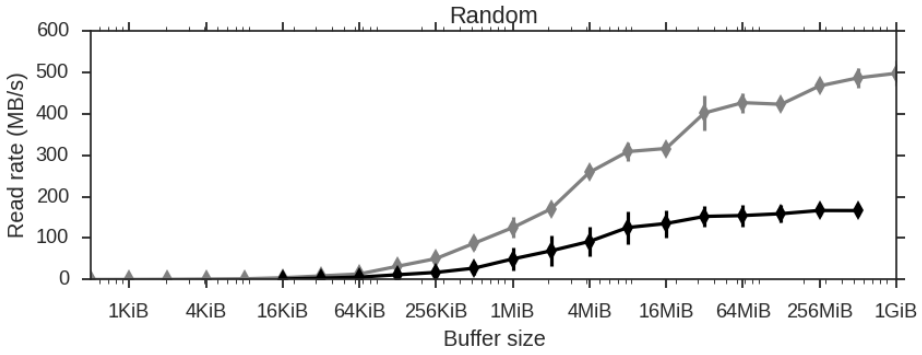
NETCDF PERFORMANCE

- Reads from NetCDF4 unchunked file using NetCDF4-python and C NetCDF library





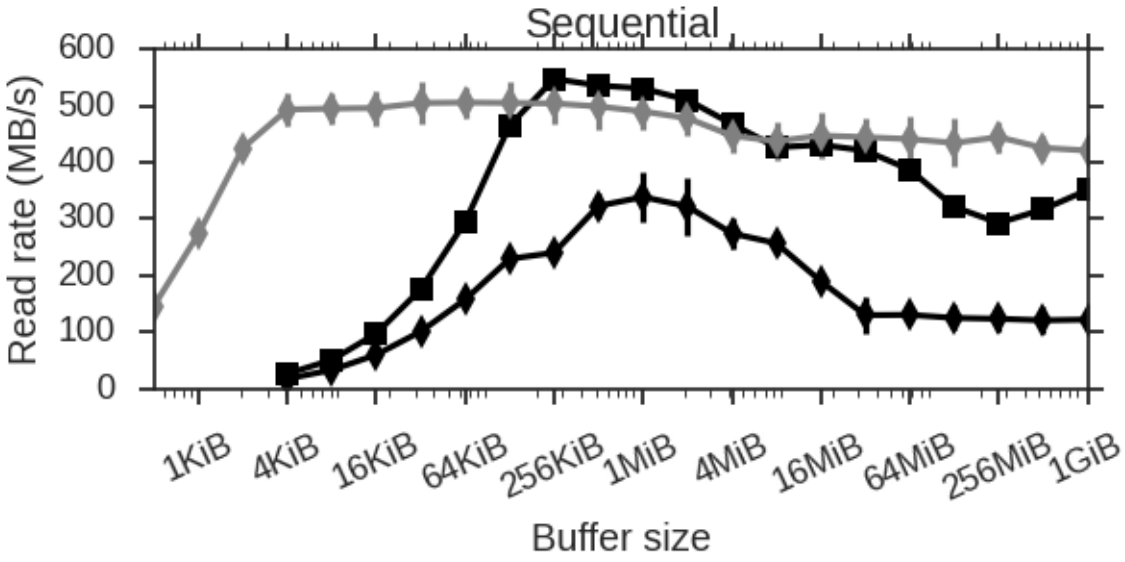
Grey – C
Black - python



University of Reading

NETCDF PERFORMANCE

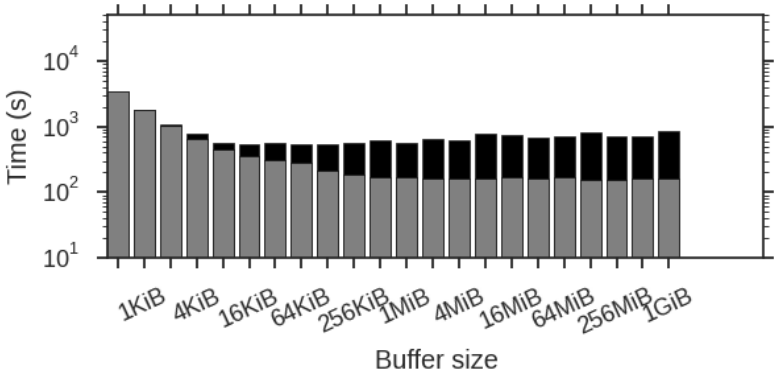
- h5netcdf library sequential reads – grey C netcdf, black diamonds netCDF4-python, black squares h5netcdf
- h5netcdf similar to plain binary reads (not shown) and C NetCDF reads at large buffer size
- Also, library always reads at least 64KiB



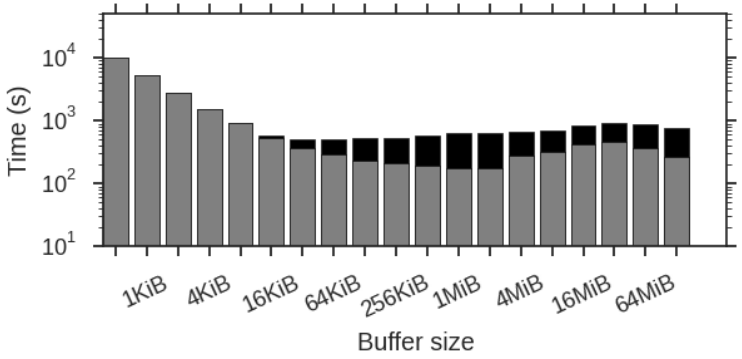
University of Reading

NETCDF PERFORMANCE

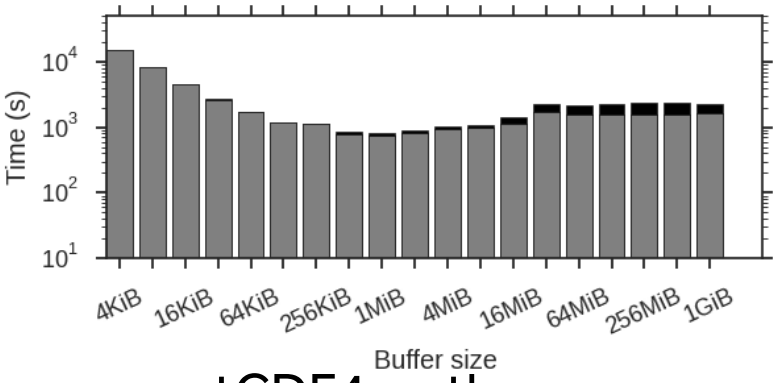
- Notable drop in performance when using NetCDF4-python
- Caused by CPU limited behaviour in the library



C NetCDF



h5netcdf



netCDF4-python

University of Reading

NETCDF PERFORMANCE

- C NetCDF4 reads similar to reading from plain binary file
  - No drop in performance purely from NetCDF4
- Notable drop in performance when using NetCDF4-python
  - Caused by CPU limited behaviour in the library
- Also, Python libraries always reads at least 64KiB from NetCDF4 files

University of Reading

CONCLUSIONS

- How much do read sizes affect read rate?
  - for netCDF4-python a lot - could be the difference between hours and days for analysis script
  - For other only below 4KiB
- How much does read pattern affect read rate?
  - Significantly – read ahead plays a large effect
- How much does using Python affect read rate?
  - Not significantly – slight drop after 16MiB
- How much does using NetCDF4 files affect read rate?
  - For C not at all
  - Within netCDF4-python library significant – but due to library

University of Reading

FUTURE WORK

- Run NetCDF4 tests on Lustre and GPFS platforms
- Quantify the effect of chunking and compression in NetCDF4 files for typical analysis script
- Parallel reads using netCDF4-python on parallel file system