I/O Performance Benchmarking and Investigation on Multiple HPC Architectures

B. Lawrencea, C. Maynardb, A. Turnerc, X. Guoc, D. Sloan-Murphyc

aUniversity of Reading, United Kingdom

bMet Office, United Kingdom

cEdinburgh Parallel Computing Centre, University of Edinburgh, United Kingdom

Abstract

I/O performance plays a key role in many scientific simulations and the bottleneck of I/O is an important challenge to solve towards Exascale computing. It is therefore necessary for CoEs and scientific communities with high I/O requirements to understand the usage pattern of existing HPC systems and applications to be suitably equipped to make informed plans for the future.

Theoretical performance numbers assume a clean formatted file system with no contention from other users. Obviously, when used in full production, this level of performance will not be attained.

One of the goals of this paper is to provide insight into the performance of the file systems in production. To answer questions such as: What is the maximum performance actually experienced? What variation in performance do users experience?

To that end, this paper details the I/O performance of multiple HPC architectures through testing a set of selected I/O benchmarks and representative scientific applications. Results are presented from the following systems:

**ARCHER**: the UK national supercomputing service, using a Cray Sonexion Lustre file system.

**COSMA**: a UK DiRAC resource, using a DDN implementation of the IBM GPFS file system.

**UK-RDF DAC**: the Data Analytic Cluster attached to the UK Research Data Facility, using DDN GPFS.

**JASMIN**: a data analysis cluster delivered by the STFC, using \*\*\*Confirm implementation\*\*\* GPFS.

\*\*\*Description of scientific application machine\*\*\*

In the Lustre case, a range of stripe counts and sizes are tested. GPFS provides less scope for user tuning, hence figures are given under the default configuration.

We benchmark a simple, EPCC-produced MPI-IO parallel application, given the name benchio, which writes a three-dimensional distributed dataset to a single shared file. On supported systems, we further measure and compare the performance of HDF5 and NetCDF equivalent implementations.

\*\*\*Description of scientific applications from ESiWACE and Met Office\*\*\*

We find \*\*\*analysis and conclusions\*\*\*

# HPC Systems

## ARCHER

ARCHER is a Cray XC30-based system and the current UK National Supercomputing Service. The /work file systems on ARCHER use the Lustre technology in the form of Sonexion parallel file system appliances. The theoretical sustained performance (in terms of bandwidth) of Sonexion Lustre file systems is determined by the number of SSUs (Scalable Storage Units) that make up the file system. ARCHER has four Sonexion file systems:

* fs1: 1 SSU, theoretical sustained = 5 GB/s
* fs2: 6 SSU, theoretical sustained = 30 GB/s
* fs3: 6 SSU, theoretical sustained = 30 GB/s
* fs4: 7 SSU, theoretical sustained = 35 GB/s

## COSMA

The Durham-based Cosmology Machine (COSMA) is one of the five systems making up the UK DiRAC facility. Its file systems use the IBM General Parallel File System (GPFS) implemented on two DDN SD12K storage controllers. The theoretical maximum performance is \*\*\*GB/s\*\*\*

## UK-RDF DAC

The UK Research Data Facility (UK-RDF) is a high volume file storage service collocated with ARCHER. Attached to it is the Data Analytic Cluster (DAC), a system for facilitating the analysis of data held at the RDF. The file system is also a DDN GPFS installation and is based on seven DDN 12K couplets. Separate metadata storage is on NetApp EF550/EF540 arrays populated with SSD drives. Three file systems are available to users:

* gpfs1: 6.4 PB storage, mounted as /nerc
* gpfs2: 4.4 PB storage, mounted as /epsrc
* gpfs3: 1.5 PB storage, mounted as /general

All DAC nodes have direct Infiniband connections to the RDF drives with a maximum theoretical performance of 7 GB/s.

## JASMIN

The Joint Analysis System (JASMIN) is an STFC-delivered service providing computing infrastructure for big data analysis. \*\*\*Also GPFS. Implementation and theoretical max details\*\*\*

## \*\*\*Scientific Application Machine\*\*\*

# Parallel I/O benchmark: benchio

The parallel I/O performance of the HPC systems was evaluated by the *benchio* application developed at EPCC. This was chosen ahead of the popular IOR benchmark for a number of reasons:

* The parallel I/O decomposition can be varied to better model actual user applications. IOR uses an extremely simplistic 1D data decomposition (Figure 1) that does not model user codes and does not test the performance of MPI-IO collective operations that are key to real performance.
* The IOR code is very opaque, this makes it very difficult to draw useful conclusions as to what variations in performance are due to.
* benchio is also able to evaluate the performance of HDF5 and NetCDF, two libraries that support parallel I/O and are commonly used by user communities on many HPC services.

The benchio source code is Open Source and is available on GitHub[1].

Figure 1. IOR data layout: simple sequential



The benchio application measures write bandwidth to a single shared file for a given problem size per processor (weak scaling), i.e. the size of the output file scales with the number of processors.

The test data is a series of double precision floating point numbers held in a 3D array and shared over processes in a 3D block decomposition (see Figure 2 and Figure 3). Halos have been added to all dimensions of the local arrays to better approximate the layout of a “real-world” scientific application. By default these local arrays are of size 1283.

Figure 2. benchio data layout: 3D strided, P2 behind P0

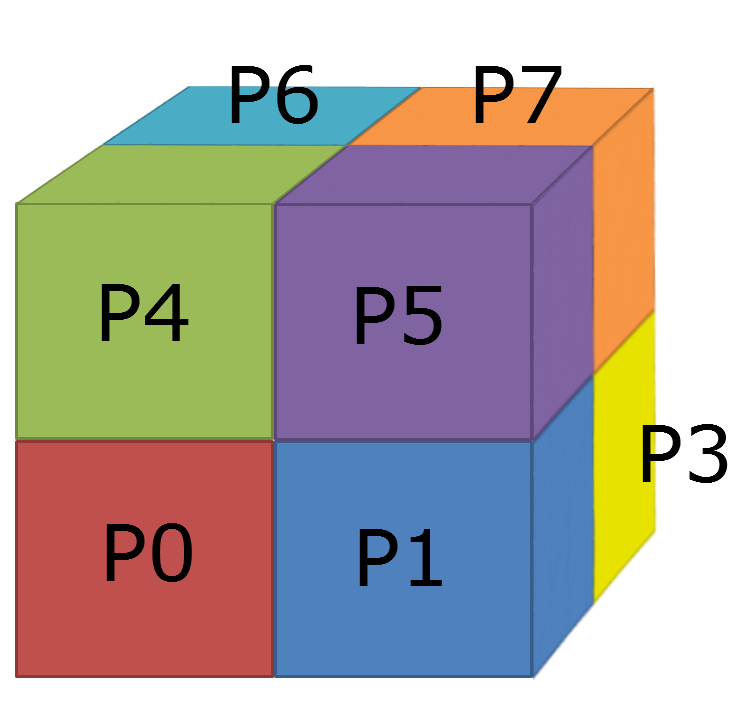
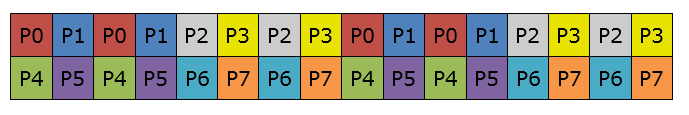


Figure . benchio data layout: example 2D decomposition, 2x2x2 grid per processor. Equivalent to layout of output file. Note: data is entirely contiguous and only split into two rows in this figure for legibility; data is not a 2x16 array



# Results

With benchio, each test is repeated a minimum of ten times and the maximum, minimum and average bandwidth reported. As I/O is a shared resource on all measured machines, and therefore subject to contention from other users, the maximum attained bandwidth is considered to be most representative of capabilities of a system. We therefore present this value in all following results, unless otherwise indicated.

## ARCHER Performance

Using the default Lustre settings on ARCHER:

* Stripe size: 1 MiB
* Number of stripes: 4

we see the performance shown in Figure 4 and listed in Table 1.

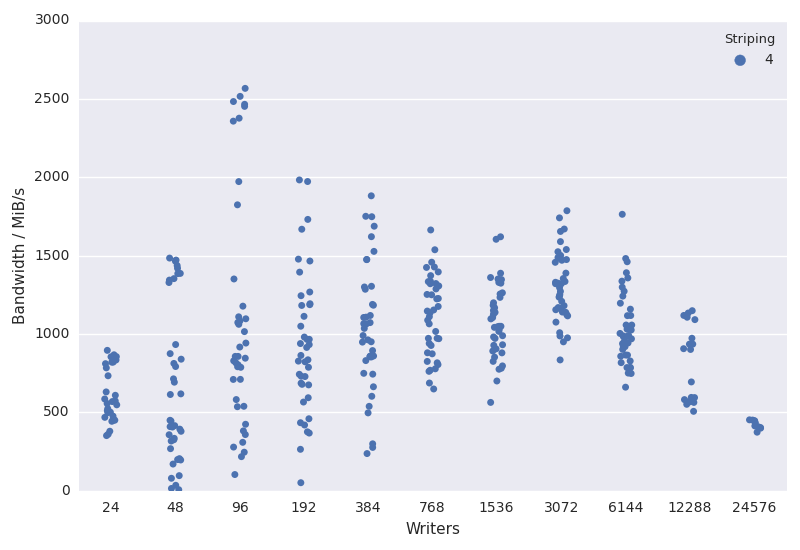


Figure 4. ARCHER MPI-IO default striping (4). A random jitter is applied to the x-axis to better illustrate clusters of similar performance.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Write Bandwidth (MiB/s)** | | | |  |
| **Writers** | **Min.** | **Median** | **Max.** | **Mean** | **Count** |
| 24 | 733.939 | 827.945 | 896.015 | 828.069 | 10 |
| 48 | 1328.688 | 1403.117 | 1484.661 | 1407.661 | 10 |
| 96 | 1351.340 | 2414.556 | 2567.143 | 2236.714 | 10 |
| 192 | 1113.092 | 1471.873 | 1982.988 | 1526.920 | 10 |
| 384 | 1061.639 | 1548.425 | 1881.732 | 1508.003 | 10 |
| 768 | 970.448 | 1270.291 | 1663.967 | 1270.813 | 10 |
| 1536 | 880.089 | 1279.216 | 1620.391 | 1241.438 | 10 |
| 3072 | 1008.584 | 1202.115 | 1475.161 | 1242.080 | 10 |
| 6144 | 817.562 | 1044.328 | 1158.771 | 1008.712 | 10 |
| 12288 | 901.767 | 1032.680 | 1149.057 | 1025.447 | 10 |
| 24576 | 374.109 | 423.413 | 452.824 | 422.868 | 10 |

Table 1. ARCHER MPI-IO default striping (4) raw data.

Using the default stripe settings on ARCHER, the maximum write performance that can be achieved is just over 2,500 MiB/s, well short of the theoretical sustained performance of 30,000 MiB/s.

#### Lustre Tuning

As described in Parallel I/O Performance on ARCHER[1], to get the best parallel write performance for a single-shared file case we must use as many stripes as possible. This is achieved on Lustre by setting the striping to “-1” which stripes over all available OSTs. We repeated the benchmarks with:

* Stripe size: 1 MiB
* Number of stripes: -1 (corresponds to 48 on fs3)

The performance for this configuration is shown in Figure 5 and Table 2.

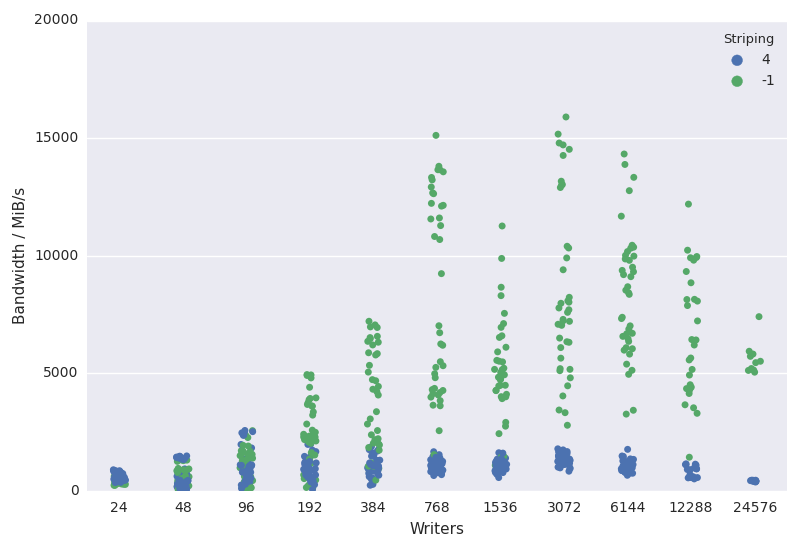


Figure 5. ARCHER MPI-IO maximum striping (-1). Default striping of 4 is plotted for comparison.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Write Bandwidth (MiB/s)** | | | |  |
| **Writers** | **Min.** | **Median** | **Max.** | **Mean** | **Count** |
| 24 | 576.868 | 612.247 | 615.717 | 605.801 | 10 |
| 48 | 1252.985 | 1318.389 | 1355.754 | 1313.825 | 10 |
| 96 | 230.472 | 1986.700 | 2559.369 | 1666.729 | 10 |
| 192 | 2284.795 | 4138.974 | 4943.626 | 4109.961 | 10 |
| 384 | 4073.371 | 5852.313 | 6971.013 | 5790.246 | 10 |
| 768 | 9235.570 | 12166.017 | 13222.881 | 11886.082 | 10 |
| 1536 | 2749.108 | 4645.424 | 5909.421 | 4561.950 | 10 |
| 3072 | 3323.747 | 6215.290 | 9904.133 | 6139.148 | 10 |
| 6144 | 5125.016 | 6358.722 | 9316.783 | 6918.379 | 10 |
| 12288 | 1429.390 | 9859.596 | 12192.030 | 8969.973 | 10 |
| 24576 | 5045.687 | 5480.116 | 7406.748 | 5633.765 | 10 |

Table 2. ARCHER MPI-IO maximum striping (-1) raw data.

When using the maximum number of stripes, we see much improved performance (compared to the default stripe count of 4) with a maximum write bandwidth of over 13,000 MiB/s with 768 cores (32 nodes) writing simultaneously. This is still less than 50% of the advertised sustained bandwidth of 30,000 GiB/s for this file system.

The experiments were then repeated with larger Lustre stripe sizes:

* Stripe sizes: 4 MiB and 8 MiB
* Number of stripes: -1 and 4

Performance is given in Figure 6 and Figure 7 with the data from the default 1 MiB configuration plotted for comparison.

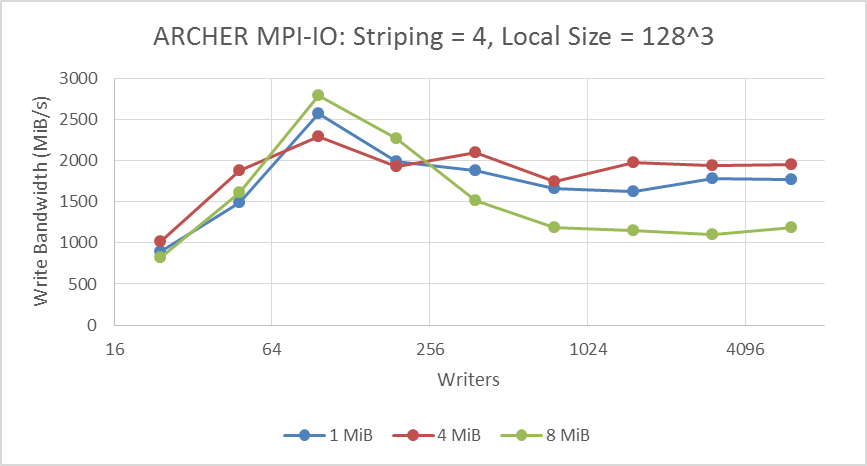


Figure 6. ARCHER stripe size performance, default stripe count

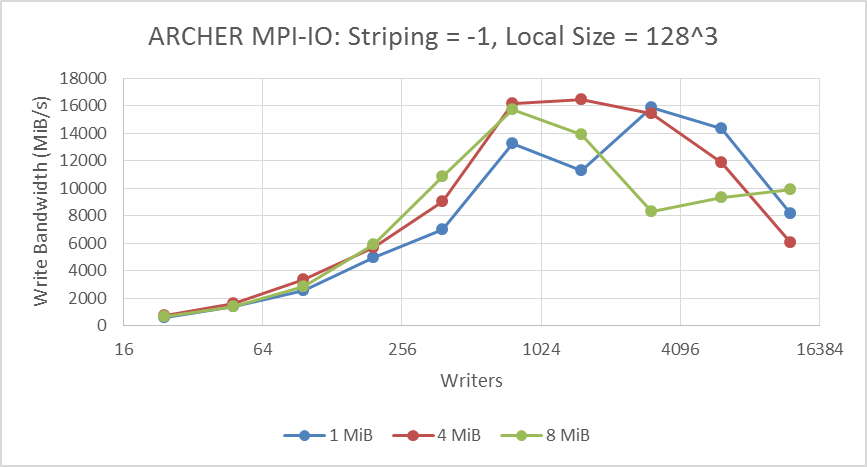


Figure 7. ARCHER stripe size performance, maximum stripe count

\*\*\*Analysis\*\*\*

#### Data Size

All prior experiments were performed with the default local data array of 1283 double precision values. We expected that the benefits of larger stripe sizes would be more pronounced with greater volumes of data so repeated the above tests with an increased array size of 2563 values. Results are given in Figure 8 and Figure 9.

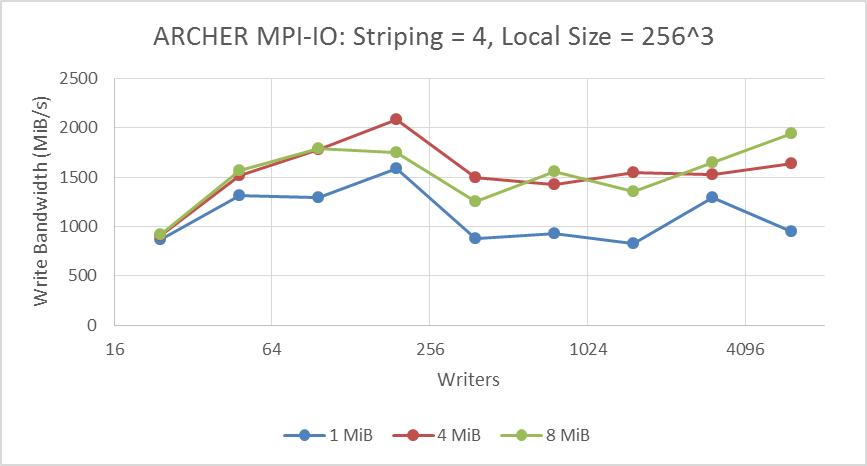


Figure 8. ARCHER large local arrays bandwidth, default stripe count

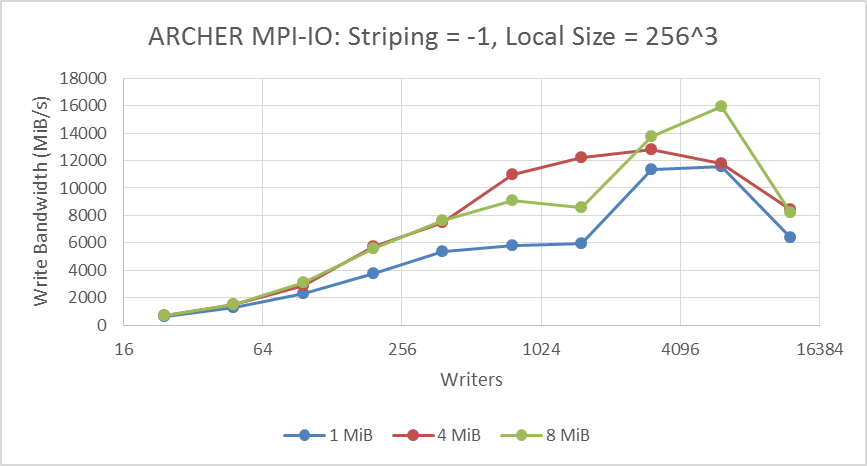


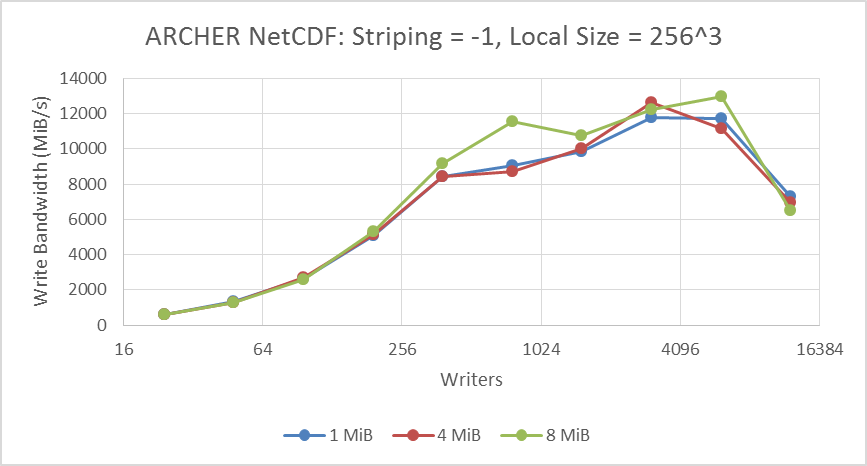
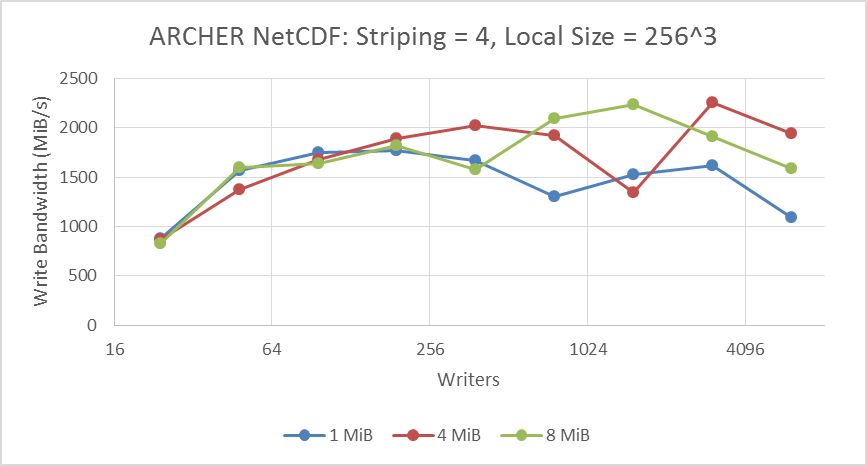
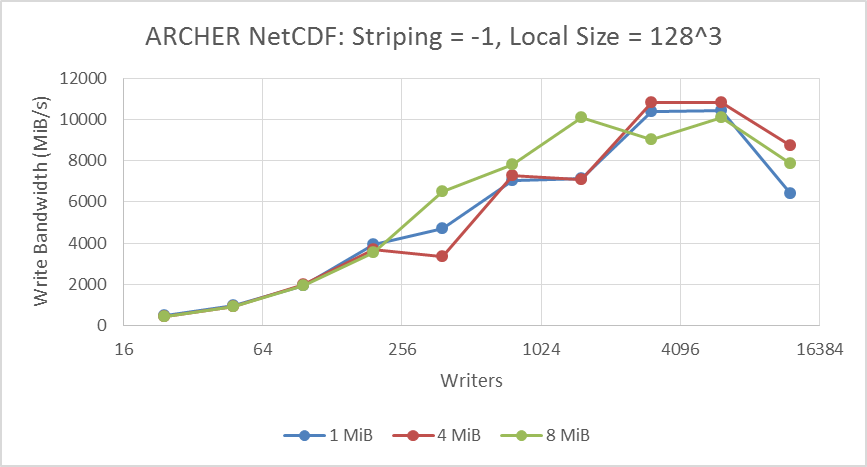
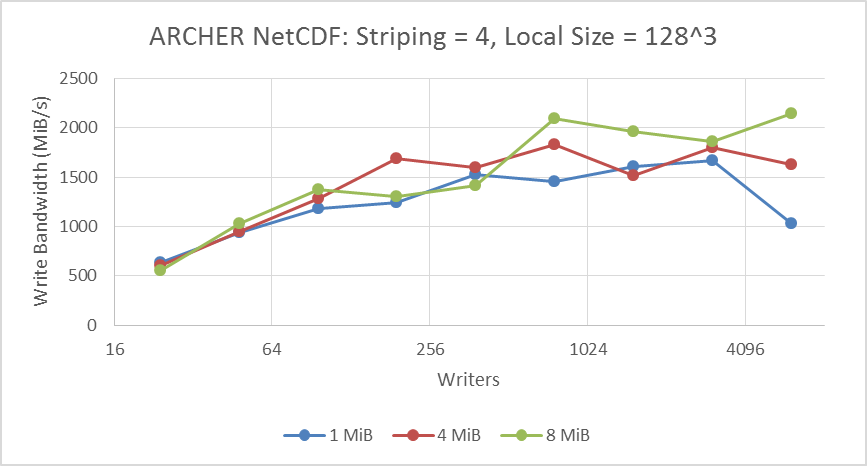
Figure 9. ARCHER large local arrays bandwidth, maximum stripe count

\*\*\*MPI-IO conclusions: best configuration is X dependent on data layout Y. Do see better performance at large stripe sizes here\*\*\*

#### NetCDF Performance

Optimised installations of NetCDF, backed by parallel HDF5, are provided by Cray as part of the operating system on ARCHER. At time of writing, the default version of this cray-netcdf-hdf5parallel module is 4.3.3.1. However, it was found to give poor performance, failing to demonstrate scalability and instead reaching a peak bandwidth of approximately 1 GiB/s regardless of number of writers. We therefore used the more recent NetCDF version 4.4.0, which scales as expected, for all benchmarks and would recommend 4.3.3.1 and below be avoided by users for performance reasons.

\*\*\*Graph analysis\*\*\*



#### HDF5 Performance

As with NetCDF, Cray provide the HDF5 parallel libraries on ARCHER. Similar performance limitations to NetCDF 4.3.3.1 were observed with HDF5 tests however they persisted with all system-installed versions of the library, from the default 1.8.14 to the most current 1.10.0.

Application profiling of benchio found the majority of time spent is in function *MPI\_File\_set\_size()*, called within the HDF5 library from the user-level *H5Fclose()* routine. Discussions with Cray revealed this is a known bug specific to the combination of HDF5 with Lustre file systems.

An *MPI\_File\_set\_size()*, on a Linux platform like ARCHER, eventually calls the POSIX operation: *ftruncate()*. This has an unfavourable interaction with the locking for the series of metadata communications the HDF5 library makes during a file close. In practice, this leads to relatively long close times of tens of seconds and hence the lack of scalability observed.

The HDF5 developers have noted this behaviour in the past where it manifested in the H5Fflush() routine, used for flushing write buffers associated with a file to disk: “when operating in a parallel application, this operation resulted in a call to MPI\_File\_set\_size, which currently has very poor performance characteristics on Lustre file systems. Because an HDF5 file’s size is not required to be accurately set until the file is closed, this operation was removed from H5Fflush and added to the code for closing a file”**Error! Reference source not found.** hence leading to the behaviour observed in *H5Fclose().*

Cray discussions on this bug are on-going and, at present, no known work-around or mitigation is provided for end users. The recommendation for CoEs is to be aware of this interaction and inform scientific communities as the issue is observed.

#### Impact of System Load

To better understand the impact of file system contention, we simulated degrees of load by running multiple instances of the benchio MPI-IO test in parallel. Figure 10 shows the aggregate mean performance of one, two and four benchio instances writing concurrently to independent files.

Note that mean performance is presented in this instance as, on occasion, test timing was such that a single benchio instance would be performing I/O while the others were in a setup phase or otherwise stalled. The maximum bandwidth is therefore equivalent to the single instance case and not a representative value for this test.

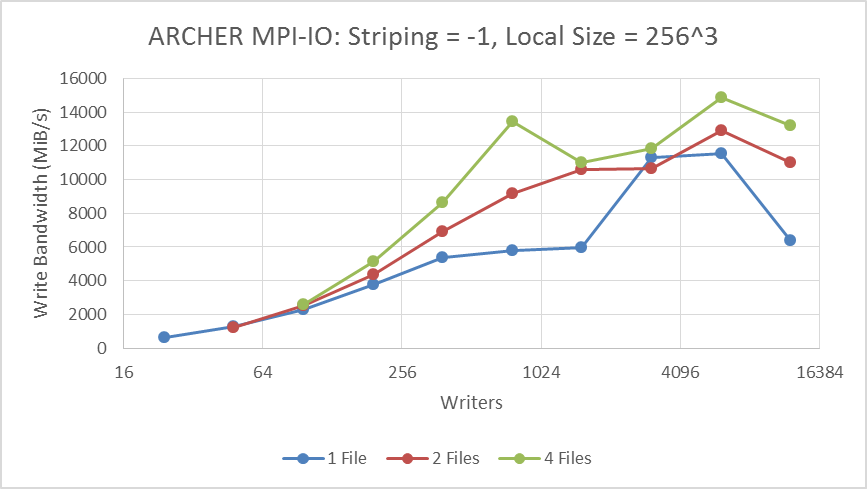


Figure 10. Effect of I/O load on ARCHER

From the similarities in the data trends, we see that bandwidth is on average divided equally between writers. e.g. the aggregate bandwidth of two benchio instances, each with 24 writers putting data to independent files, is roughly equivalent to the bandwidth of a single instance with 48 writers.

## COSMA Performance

\*\*\*Graphs\*\*\*

## UK-RDF DAC Performance

\*\*\*Graphs\*\*\*

## JASMIN Performance

\*\*\*Graphs\*\*\*

## Scientific Application Benchmarks: X and Y

\*\*\*Met Office system and application details, etc.\*\*\*

# Conclusions

\*\*\*Should expect around 50% of theoretical max in all cases, avoid HDF5 on Lustre, don’t use NetCDF 4.3.x, etc.\*\*\*

Text

Text

Text

Text

Text

Text

Text

Text

Text

References

1. EPCCed/benchio: EPCC I/O benchmarking applications, <https://github.com/EPCCed/benchio>, retrieved 01 Nov 2016
2. David Henty, Adrian Jackson, Charles Moulinec, Vendel Szeremi: Performance of Parallel IO on ARCHER Version 1.1, <http://www.archer.ac.uk/documentation/white-papers/parallelIO/ARCHER_wp_parallelIO.pdf>, retrieved 01 Nov 2016
3. Mark Howison, Quincey Koziol, David Knaak, John Mainzer, John Shalf: Tuning HDF5 for Lustre File Systems, https://support.hdfgroup.org/pubs/papers/howison\_hdf5\_lustre\_iasds2010.pdf , retrieved 03 Nov 2016

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU’s Horizon 2020 research and innovation programme (2014-2020) under grant agreement 653838.

The authors would like to thank Harvey Richardson of Cray Inc. for his invaluable advice on the ARCHER file systems and software.