Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Parallel I/O Performance Benchmarking and Investigation on Multiple HPC Architectures

B. Lawrence[a], C. Maynard[b], A. Turner[c], X. Guo[c], D. Sloan-Murphy[c]

[a]*University of Reading, United Kingdom*
[b]*Met Office, United Kingdom*
[c]*Edinburgh Parallel Computing Centre, University of Edinburgh, United Kingdom*

Abstract

Solving the bottleneck of I/O is key in the move towards exascale computing. Research communities must be informed of the I/O performance of existing resources in order to make reasonable decisions for the future. This paper therefore presents benchmarks for the write capabilities of the ARCHER, COSMA, UK-RDF DAC, and JASMIN HPC systems, using MPI-IO and, in selected cases, the HDF5 and NetCDF parallel libraries.

We find a reasonable expectation is for approximately 50% of the theoretical system maximum bandwidth to be attainable in practice. Contention is shown to have a dramatic effect on performance. MPI-IO, HDF5 and NetCDF are found to scale similarly but the high-level libraries introduce a small amount of performance overhead.

For the Lustre file system, on a single shared file, maximum performance is found by maximising the stripe count and matching the individual stripe size to the magnitude of I/O operation performed. HDF5 is discovered to scale poorly on Lustre due to an unfavourable interaction with the *H5Fclose()* routine.

Introduction

Parallel I/O performance plays a key role in many high performance computing (HPC) applications and I/O bottlenecks are an important challenge to understand and, where possible, eliminate on both current, petascale resources and looking forward to exascale computing[1]. It is therefore necessary for research communities with high I/O requirements to understand the parallel I/O performance of existing HPC systems and applications to be suitably equipped to make informed plans for future procurements and software development projects. The results of this work are of particular relevance to the ESiWACE Centre of Excellence (CoE), as the originators of this project, but, given the ubiquity of I/O in HPC domains, the findings will be of interest to most researchers and members of the European scientific community.

Theoretical performance numbers for parallel file systems are usually easily available but are of limited use as they assume a clean formatted file system with no contention from other users. Obviously, when used in full production, this level of performance will not usually be attained.

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

To this end, we detail here the parallel I/O performance of multiple HPC architectures through testing a set of selected I/O benchmarks. Results are presented from the following systems:

- **ARCHER:** the UK national supercomputing service, with a Cray Sonexion Lustre file system.
- **COSMA:** one of the DiRAC UK HPC resources, using a DDN implementation of the IBM GPFS file system.
- **UK-RDF DAC:** the Data Analytic Cluster attached to the UK Research Data Facility, also using DDN GPFS.
- **JASMIN:** a data analysis cluster delivered by the STFC, using the Panasas parallel file system.

We run benchio, a parallel benchmarking application which writes a three-dimensional distributed dataset to a single shared file. On all systems, we measure MPI-IO performance and, in selected cases, compare this with HDF5 and NetCDF equivalent implementations.

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

This document is structured as follows: in the subsequent section, we provide detailed specifications on the four chosen benchmark systems and their file systems. We then present our benchio application, highlighting the contrast between its data layout and the layout used by more traditional benchmarks. Results and conclusions follow, and we close by highlighting the opportunities for future work identified during the course of this project.

HPC Systems

ARCHER

ARCHER[2] is a Cray XC30-based system and the current UK National Supercomputing Service run by EPCC[3] at the University of Edinburgh[4]. 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 three Sonexion file systems available to users:

- fs2: 6 SSU, theoretical sustained = 30 GB/s
- fs3: 6 SSU, theoretical sustained = 30 GB/s
- fs4: 7 SSU, theoretical sustained = 35 GB/s

Each compute node on ARCHER has two Intel Xeon E5-2697 v2 (Ivy Bridge) processors running at 2.7 GHz containing 12 cores each, giving a total of 24 cores per node. Standard compute nodes have 64 GB of memory shared between the two processors. A set of high-memory nodes are offered with 128 GB of available memory but these are not considered in this paper.

Compute nodes are linked via the Cray Aries interconnect[5], a low-latency, high-bandwidth link giving a peak bandwidth of approximately 11,090 GB/s over the entire ARCHER machine.

COSMA

The Durham-based Cosmology Machine (COSMA)[6] is one of the five systems making up the UK DiRAC facility[7]. Its disks use the IBM General Parallel File System (GPFS) implemented on two DDN SD12K storage controllers. The theoretical maximum performance is 20 GB/s.
Each compute node on COSMA has two 2.6 GHz Intel Xeon E5-2670 CPUs with 8 cores each, i.e. 16 cores per node. 128 GB of RAM is available as standard and the interconnect between node and file system is Mellanox Infiniband FDR10.

UK-RDF DAC

The UK Research Data Facility (UK-RDF)[8] is a high-volume file storage service collocated with ARCHER. Attached to it is the Data Analytic Cluster (DAC)[9], a system for facilitating the analysis of data held at the

RDF. The file system is 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

The DAC offers two compute node configurations: standard, using two 10-core 2.20 GHz Intel Xeon E5-2660 v2 processors and 128 GB RAM; and high-memory, using four 8-core 2.13 GHz Intel Xeon E7-4830 processors and 2 TB RAM. In this paper, the standard nodes are used exclusively to model the typical use case.

All DAC nodes have direct Infiniband connections to the RDF drives with a maximum theoretical performance of 56 Gbps, or 7 GB/s.

JASMIN

The Joint Analysis System (JASMIN)[10] is an STFC-delivered service providing computing infrastructure for big data analysis.

All tests were run from the Lotus compute cluster on JASMIN on nodes with 2.6 Ghz 8-core Intel Xeon E5-2650 v2 processors and 128 GB memory. The cluster uses the Panasas parallel file system implemented via bladesets connected to compute nodes over a 10 Gbps, i.e. 1.25 GB/s, Ethernet network, the theoretical limit for performance.

Parallel I/O benchmark: benchio

The parallel I/O performance of the HPC systems was evaluated by the *benchio* application developed at EPCC. The code is Open Source and is available on GitHub[11]. It 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.
- 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.

Elaborating on the first reason listed, 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. This is supported by previous work in *Parallel IO Benchmarking*[1] which found that the optimal MPI-IO write configuration for the IOR layout is to disable collective I/O, a feature essential for achieving speeds beyond that of a few kilobytes-per-second on realistic data layouts.

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. We chose to measure write bandwidth as it is the critical consideration of scientific application I/O performance, whereas read performance is traditionally not a factor beyond the initial "one-off" cost of reading input files.

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, each of these local arrays are of size 128^3.

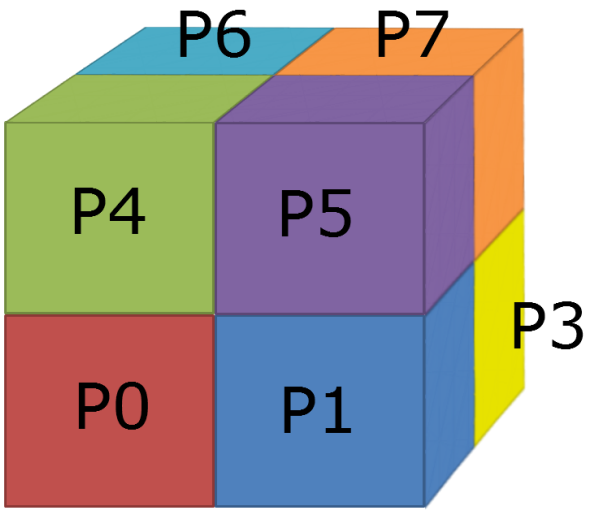

Figure 2. benchio data layout: 3D strided, P2 behind P0

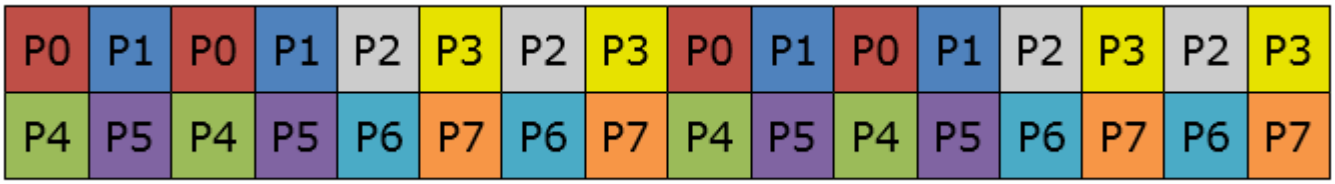

Figure 3. benchio data layout: example 3D decomposition, 2x2x2 grid per processor. Equivalent to layout of output file. Note: data is an entirely contiguous 1x32 array, split into two rows in this figure only for legibility. Contrast with the IOR parallel data layout shown in Figure 1.

Results

With benchio, each test is repeated a minimum of ten times and the maximum, minimum and mean 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. In our initial ARCHER results, we present the full range of values to demonstrate the high variance caused by user contention. However, in the results following, we present only the maximum unless otherwise indicated.

ARCHER Performance

Benchio was compiled on ARCHER with the following modules loaded:

1) modules/3.2.10.2
2) eswrap/1.3.3-1.020200.1278.0
3) switch/1.0-1.0502.57058.1.58.ari

4) craype-network-aries
5) craype/2.4.2
6) cce/8.4.1
7) cray-libsci/13.2.0
8) udreg/2.3.2-1.0502.9889.2.20.ari
9) ugni/6.0-1.0502.10245.9.9.ari
10) pmi/5.0.7-1.0000.10678.155.25.ari
11) dmapp/7.0.1-1.0502.10246.8.47.ari
12) gni-headers/4.0-1.0502.10317.9.2.ari
13) xpmem/0.1-2.0502.57015.1.15.ari
14) dvs/2.5_0.9.0-1.0502.1958.2.55.ari
15) alps/5.2.3-2.0502.9295.14.14.ari
16) rca/1.0.0-2.0502.57212.2.56.ari
17) atp/1.8.3
18) PrgEnv-cray/5.2.56
19) pbs/12.2.401.141761
20) craype-ivybridge
21) cray-mpich/7.2.6
22) packages-archer
23) bolt/0.6
24) nano/2.2.6
25) leave_time/1.0.0
26) quickstart/1.0
27) ack/2.14
28) xalt/0.6.0
29) epcc-tools/6.0
30) cray-netcdf-hdf5parallel/4.4.0
31) cray-hdf5-parallel/1.8.16

using the Cray Fortran compiler with the default compile flags.

Using the default Lustre settings on ARCHER:

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

and running on the fs3 file system, as defined above, we see the performance shown in Figure 4 and listed in Table 1. Recall that each compute node on ARCHER has 24 compute cores and that all cores per node are used when running benchio, giving 24 writers per node.

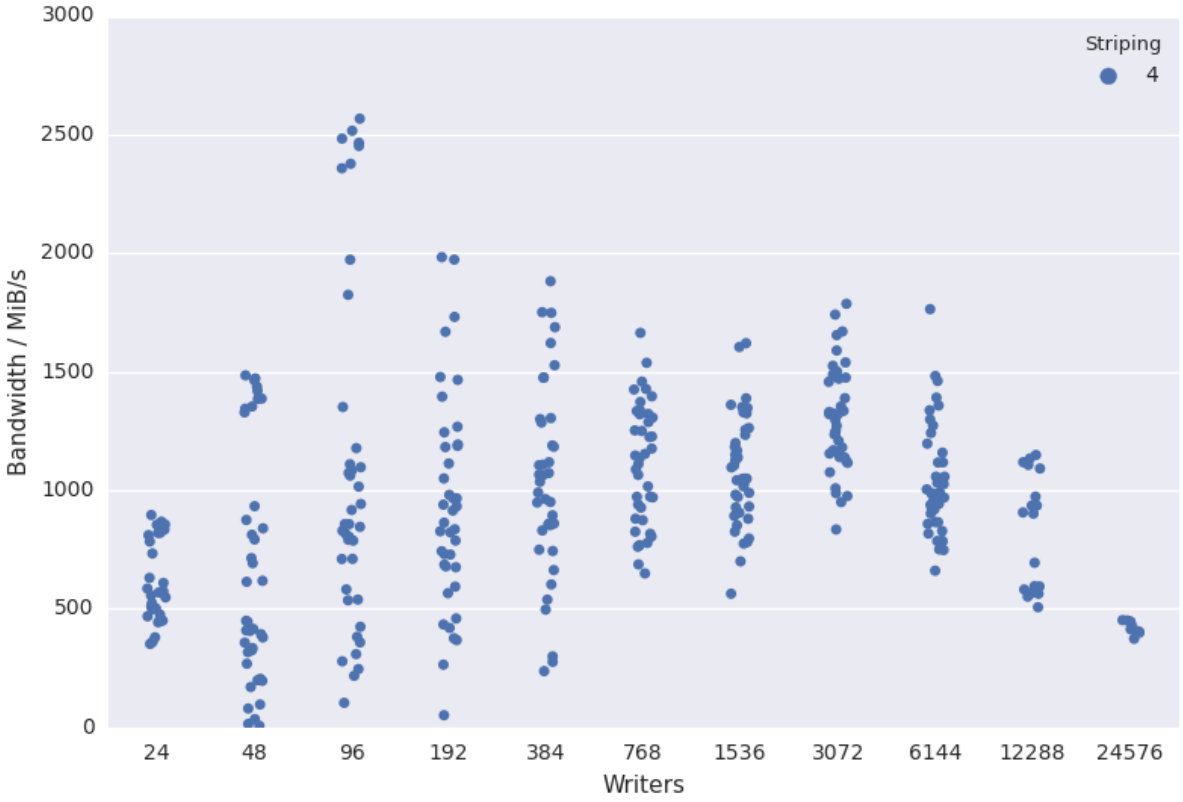

Figure 4. ARCHER MPI-IO default striping (4). A random jitter is applied to the x-axis to better illustrate clusters of similar performance.

		Write Bandwidth (MiB/s)				
Writers	**Total MiB**	**Min.**	**Median**	**Max.**	**Mean**	**Count**
24	384	352	563	896	608	30
48	768	7	448	1485	662	40
96	1536	104	858	2567	1096	40
192	3072	52	889	1983	939	40
384	6144	238	1049	1882	1042	40
768	12288	650	1141	1664	1117	40
1536	24576	564	1049	1620	1081	40
3072	49152	835	1309	1787	1307	40
6144	98304	661	986	1764	1041	40
12288	196608	507	798	1149	803	20
24576	393216	374	423	453	423	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, just 8.3% of the theoretical sustained performance of 30,000 MiB/s.

In the worst case, 48 writers give a speed of approximately 7 MiB/s, more than a factor of 200 slower than the maximum performance of near 1,500 MiB in that instance. This clearly illustrates the extreme effects file system contention from other users can have on the range of I/O performance.

Lustre Tuning

As described in *Parallel I/O Performance on ARCHER*[13], 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:

- File system: fs3
- 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.

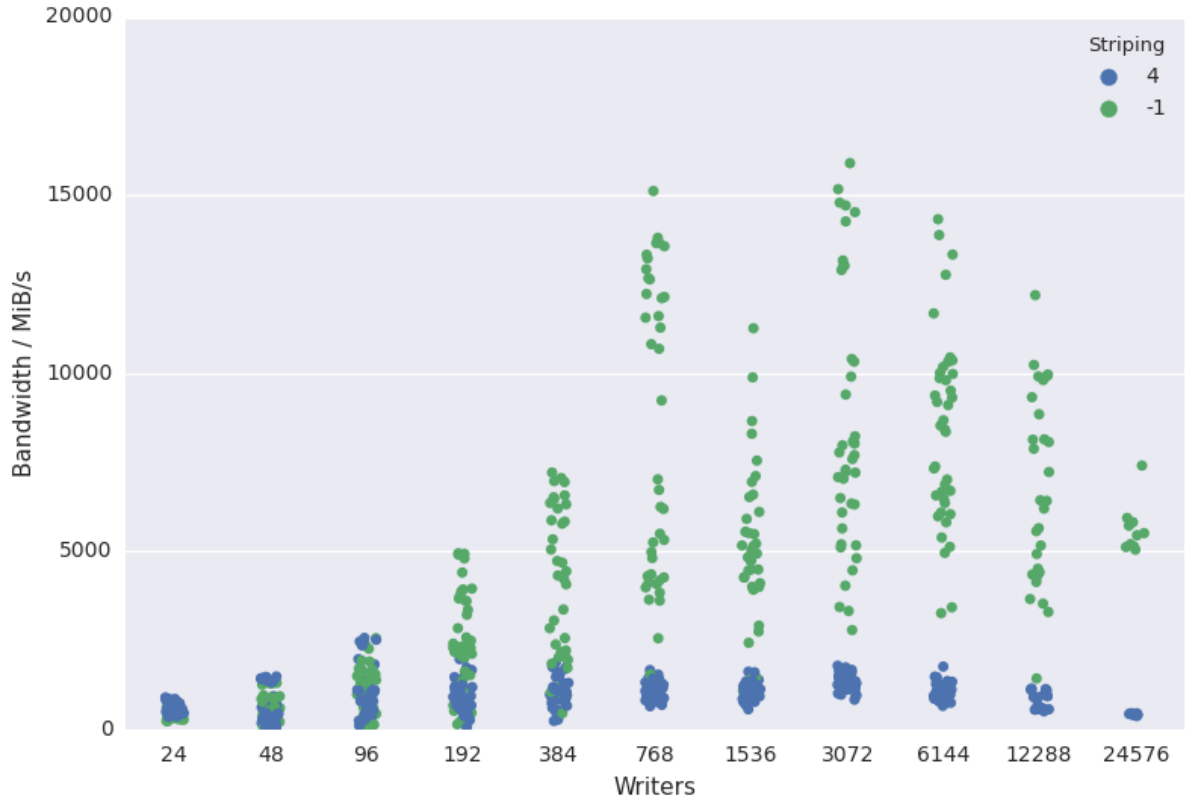

Figure 5. ARCHER MPI-IO maximum striping (-1). Default striping of 4 is plotted for comparison.

		Write Bandwidth (MiB/s)				
Writers	**Total MiB**	**Min.**	**Median**	**Max.**	**Mean**	**Count**
24	384	234	396	616	432	30
48	768	24	581	1356	694	40
96	1536	93	1289	2559	1233	40
192	3072	123	2317	4944	2547	40
384	6144	455	4145	7210	3890	40
768	12288	1541	6872	15116	8318	40
1536	24576	919	4883	11262	5050	40
3072	49152	2789	7645	15898	8547	40
6144	98304	3263	8477	14323	8371	40
12288	196608	1429	6308	12192	6598	30
24576	393216	5046	5480	7407	5634	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 slightly under 16,000 MiB/s with 3072 cores (128 nodes) writing simultaneously. This is a performance of just over 50% of the advertised sustained bandwidth of 30,000 MiB/s for this file system.

The experiments were then repeated, adjusting the size of each Lustre stripe:

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

Maximum measured performance is given in Figure 6 and Figure 7 with the data from the default 1 MiB configuration plotted for comparison. As previously stated, we plot the maximum rather than mean, median or other percentile to account for the high variance in results from contention.

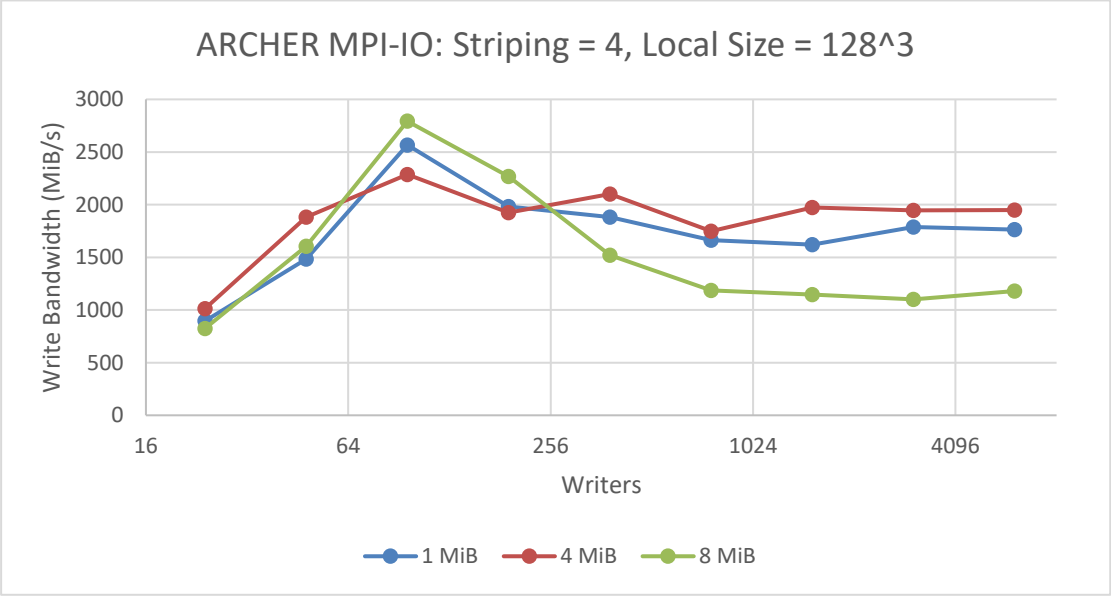

Figure 6. ARCHER stripe size performance, default stripe count

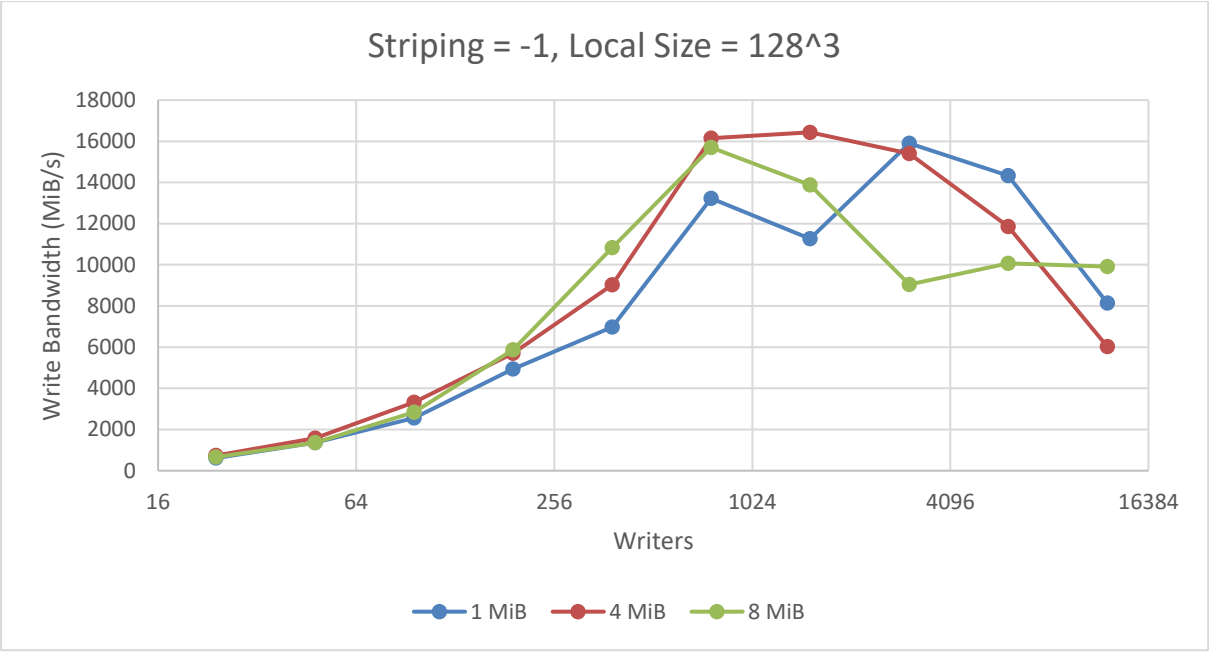

Figure 7. ARCHER stripe size performance, maximum stripe count

Stripe size was found to have a limited effect on the write performance, with the peak for all three sizes being approximately 16,000 MiB/s as before and the measured differences being in-line with the expected variance caused by file system contention. All three settings are shown to be detrimental as core counts increase beyond this performance peak, an effect attributed to increased file locking times and OST contention.

Data Size

All prior experiments were performed with the default local data array of 128^3 double precision values (16 MiB) of data per process. We expected that the benefits of larger stripe sizes would be made apparent with greater volumes of data so repeated the above tests with an increased array size of 256^3 values (128 MiB) per process. Results are given in Figure 8 and Figure 9.

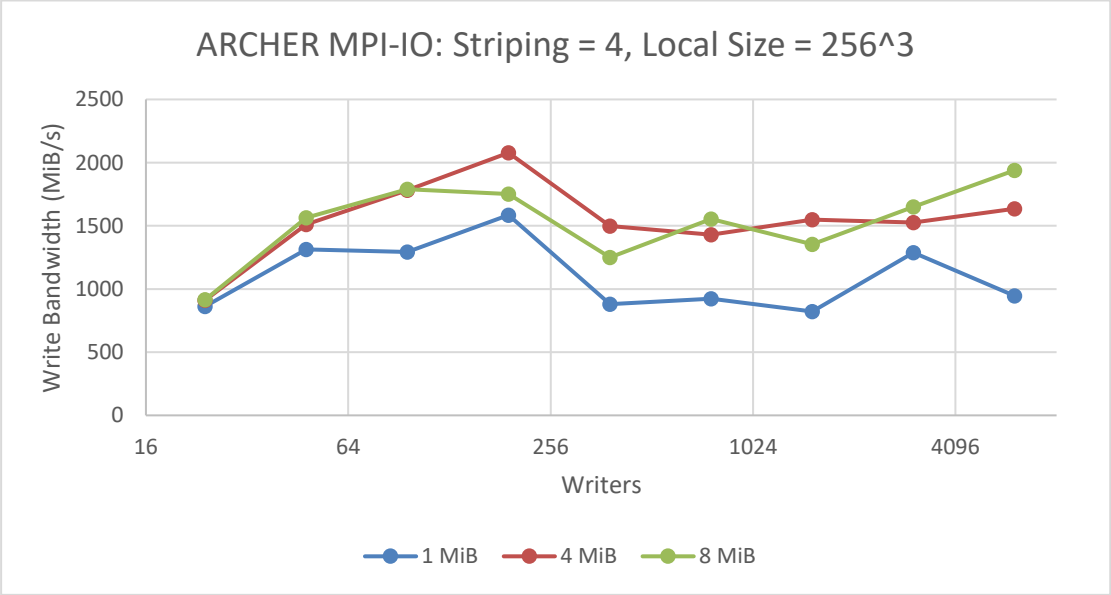

Figure 8. ARCHER large local arrays bandwidth, default stripe count

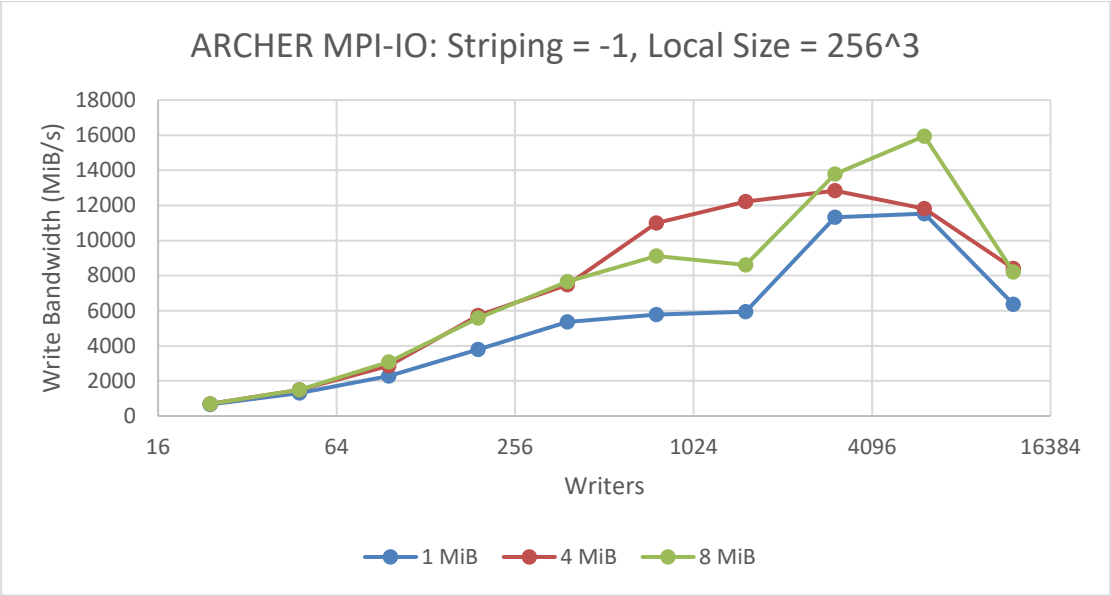

Figure 9. ARCHER large local arrays bandwidth, maximum stripe count

The larger 4 MiB and 8 MiB stripe sizes give consistently better performance than the default 1 MiB at both 4 and -1 stripe counts. Indeed 8 MiB at 6144 cores is the only configuration to achieve the apparent 16,000 MiB/s limit on ARCHER I/O while the default 1 MiB reaches less than 12,000 MiB/s.

It is apparent that stripe size configuration must be considered in conjunction with I/O operation size to attain maximum performance. In general they must match; lower volume operations should be given smaller stripe sizes, while larger operations require larger stripes.

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 or Lustre configuration. We therefore used the more recent NetCDF version 4.4.0 which scales as expected for all benchmarks and recommend to avoid the use of NetCDF versions 4.3.3.1 and below for performance reasons.

Results for version 4.4.0, repeating the stripe and array size experiments performed for MPI-IO, are plotted in Figure 10 to Figure 13.

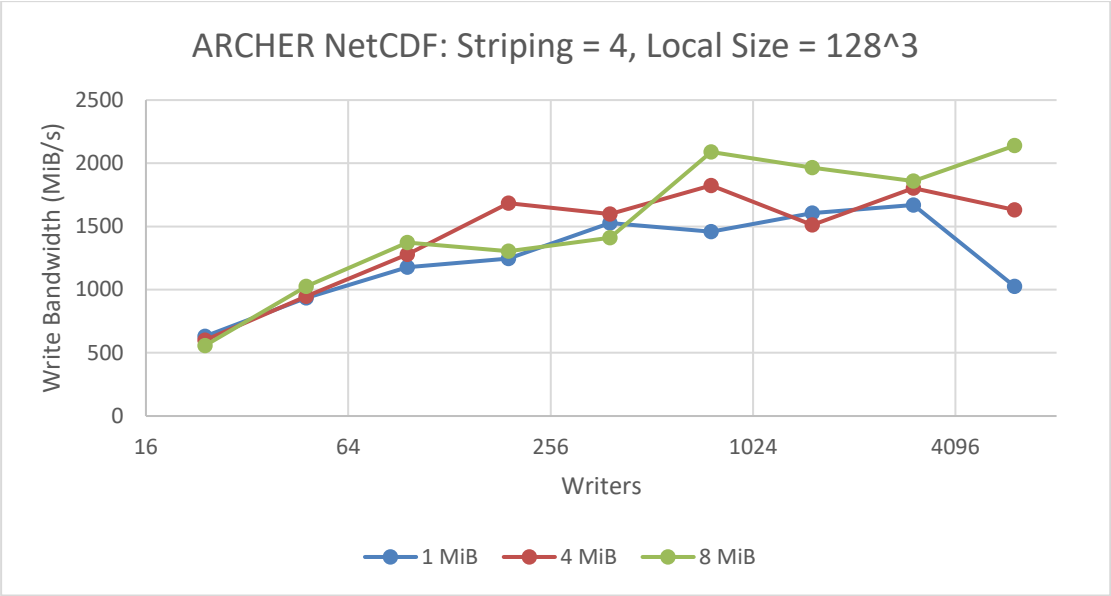

Figure 10. ARCHER NetCDF v4.4.0 performance, default striping, default array sizes

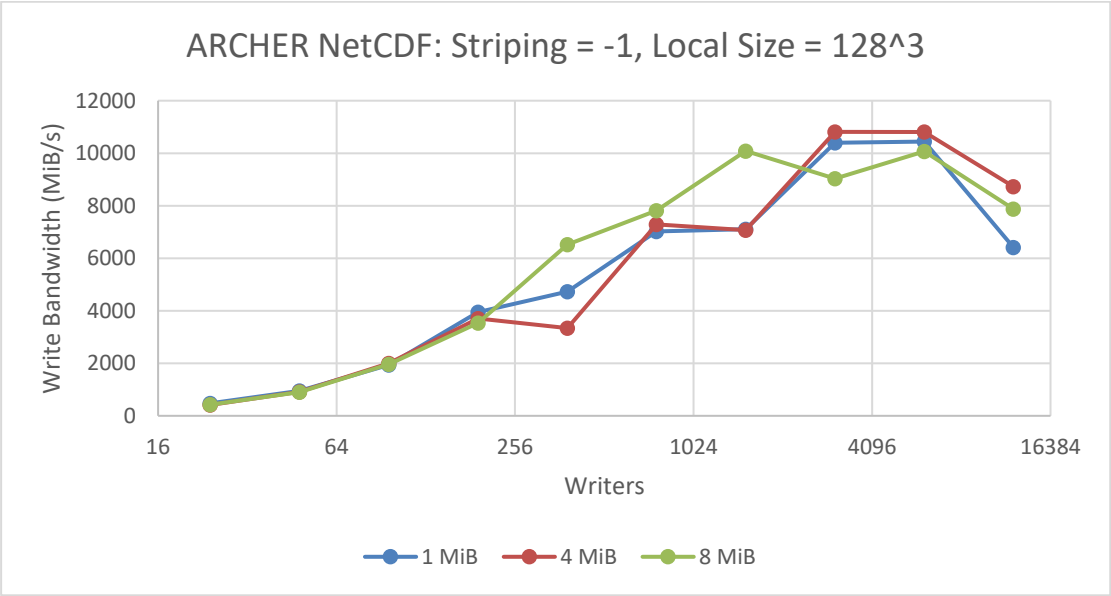

Figure 11. ARCHER NetCDF v4.4.0 performance, maximum striping, default array sizes

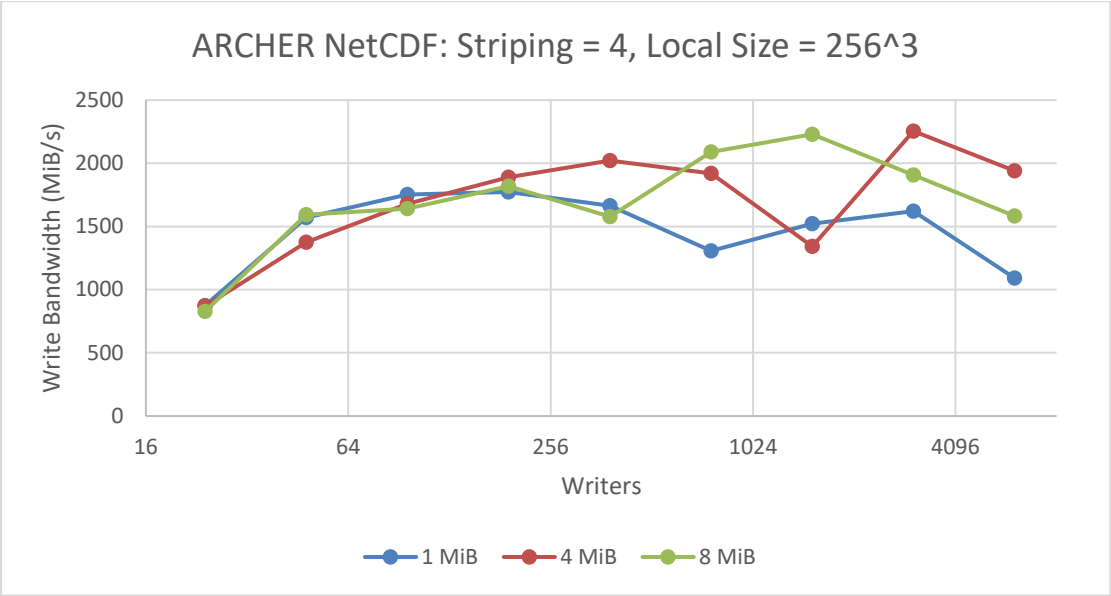

Figure 12. ARCHER NetCDF v4.4.0 performance, default striping, large arrays

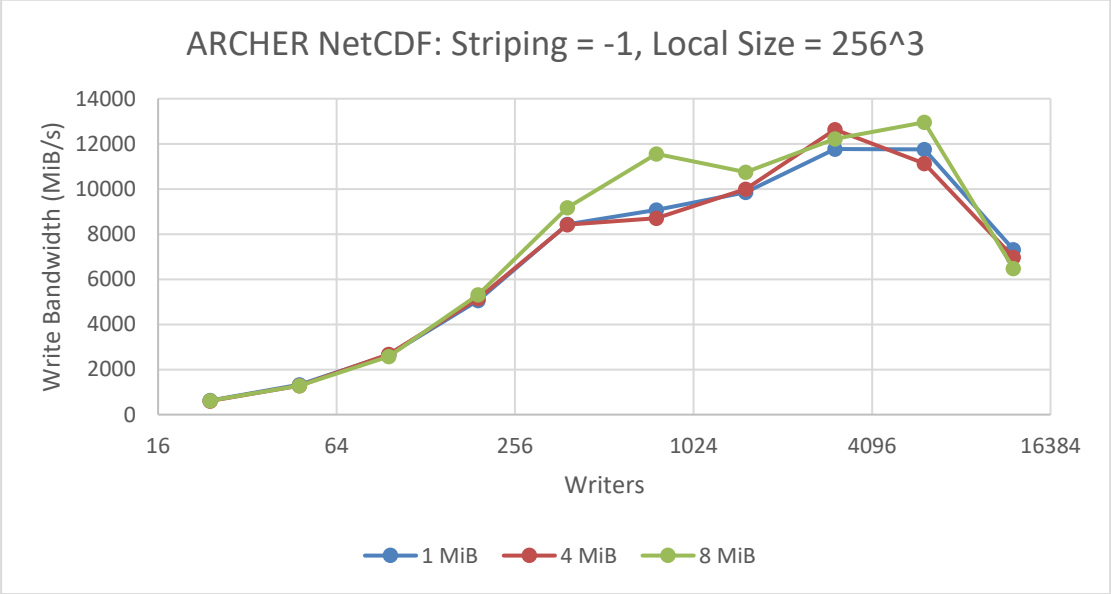

Figure 13. ARCHER NetCDF v4.4.0 performance, maximum striping, large arrays

NetCDF performance characteristics were found to be entirely similar to MPI-IO, with variations in stripe count, stripe size and local array size producing the same general trend. This is in line with expectations as NetCDF interfaces to HDF5 for its parallel implementation, which is itself based on MPI-IO.

Peak bandwidth was measured at 13,000 MiB/s, down from the 16,000 MiB/s seen with MPI-IO, i.e NetCDF achieves roughly 80% of MPI-IO performance. This is attributed to the overhead of the NetCDF/HDF5/MPI-IO stack and the additional structuring applied to NetCDF files. To verify this, we examined the write statistics recorded by MPICH, specifically those reported through the *MPICH_MPIIO_STATS* environment variable. Extracts from a simple base case – single writer, maximum striping – are given below:

MPIIO write access patterns for striped/mpiio.dat
independent writes = 0
collective writes = 24

MPIIO write access patterns for striped/hdf5.dat

independent writes = 6
collective writes = 24

MPIIO write access patterns for striped/netcdf.dat
independent writes = 10
collective writes = 24

From this, we can see the actual parallel I/O performed, the collective writes count, is identical between the three libraries, while independent writes increase with the richness of the structural and header information provided. This partially accounts for the lowered performance peak with the remaining deficit being additional time spent in library-specific functions. This last point is of particular relevance in the case of HDF5 on ARCHER, detailed in the following section.

HDF5 Performance

As with NetCDF, Cray provides several pre-installed versions of the HDF5 parallel library on ARCHER. For these library versions (from the default 1.8.14 to the most current 1.10.0), similar performance limitations as for NetCDF 4.3.3.1 were observed. Given the hierarchical nature of the libraries, we theorised that the NetCDF 4.3.3.1 limitations were in reality a manifestation of a bug in the HDF5 layer, and that NetCDF 4.4.0 circumvented the issue by following an alternate code path around the problematic library calls.

Application profiling of benchio with the HDF5 backend, to verify this theory, found the majority of compute time is spent in function *MPI_File_set_size()*, called within the HDF5 library from the user-level *H5Fclose()* routine. Discussions with Cray revealed this to indeed be a known bug specific to the combination of HDF5 with Lustre file systems.

An *MPI_File_set_size()* operation, on a Linux platform like ARCHER, eventually calls the POSIX function: *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 *H5Fflush()*, the function 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”[14] hence leading to the behaviour currently observed in *H5Fclose()*.

Cray’s investigations 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 research communities as the issue is observed.

Impact of System Load

To better understand the impact of file system contention, we simulated different degrees of load by running multiple instances of the benchio MPI-IO test in parallel. Figure 14 shows the aggregate mean performance of one, two and four benchio instances writing concurrently to independent files with the default stripe size (1 MiB).

Note that here we use aggregate mean performance, rather than maximum performance, as, in the given setup, often a single benchio instance would be performing I/O while the other instances were preparing to start, had already finished or were otherwise between iterations. The maximum bandwidth achieved during such a test is essentially the same as the maximum bandwidth when running just a single benchio instance and is therefore not representative of the impact of system load.

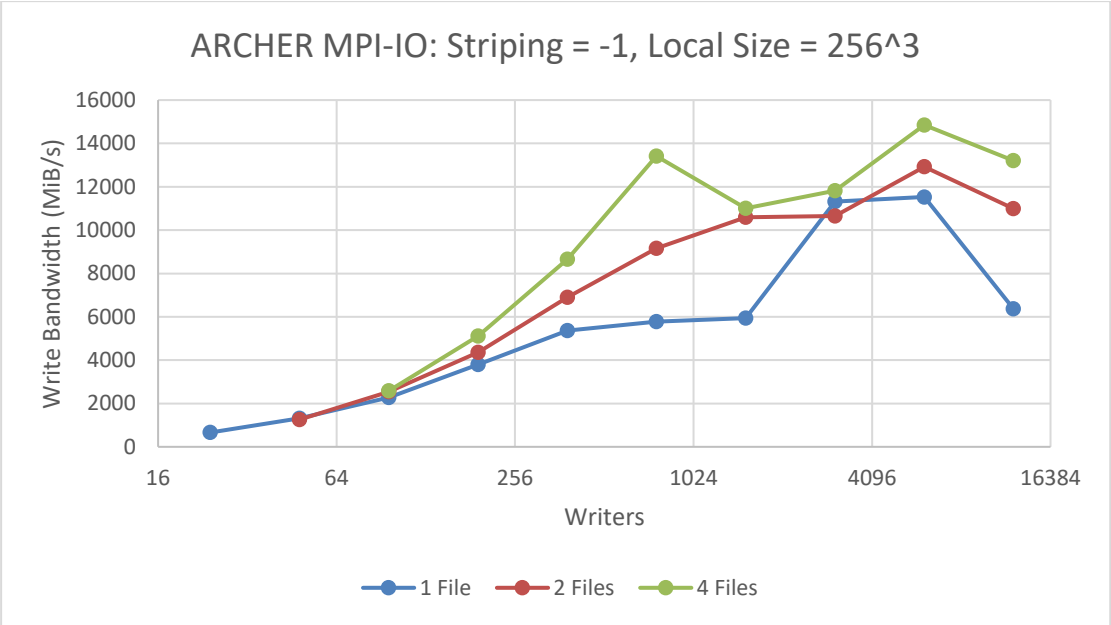

Figure 14. Effect of I/O load on ARCHER

At core counts below 96, the data trends are reasonably similar and 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. However, as number of writers increase, there is a definite trend that multiple files give better performance than a single file. This is particularly apparent in the 768 writers case where a single file sees approximately 5800 MiB/s while four files achieves near 14000 MiB/s, more than a factor of two difference. In further work, investigations into using varying numbers of files, from the current findings on a single shared file to the extreme case of a single file per process, could be done to further explore the results seen here.

COSMA Performance

The GPFS file system employed by the DiRAC COSMA service does not facilitate user tuning like Lustre. GPFS settings are fixed at installation and cannot be adjusted at run time. We therefore ran a single set of benchmarks to determine the peak bandwidth of the system, presented in Figure 15. NetCDF and HDF5 results were not gathered as they are not supported on COSMA by default.

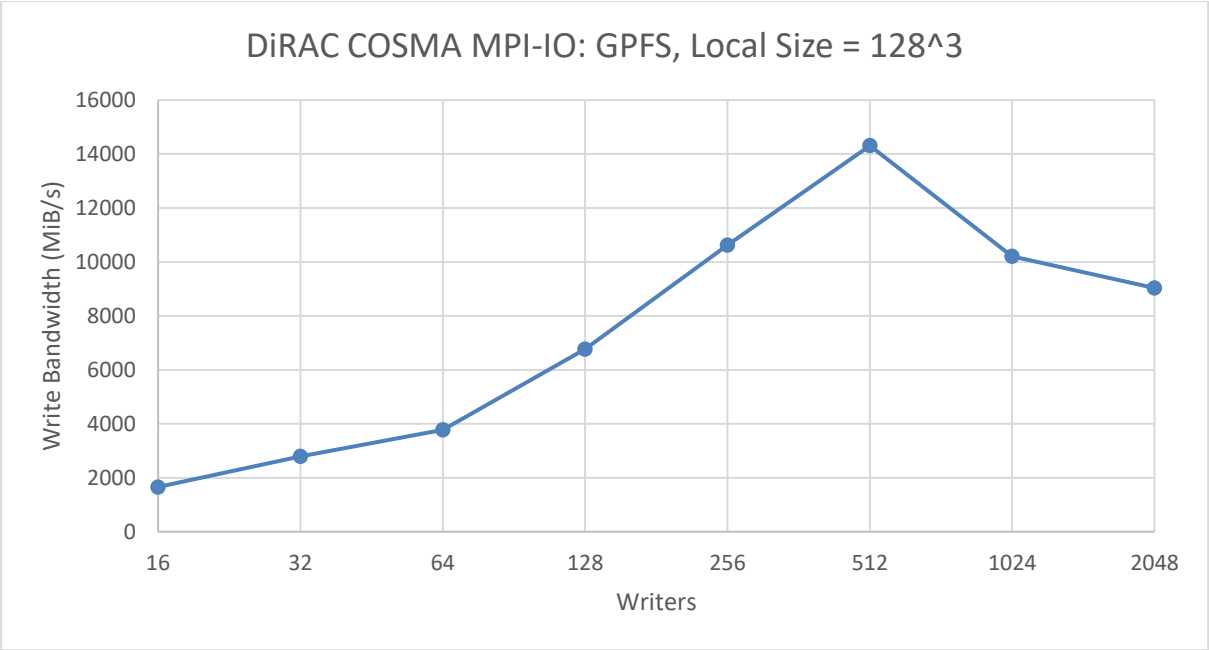

Figure 15. MPI-IO bandwidth for DiRAC COSMA

Best performance is seen at 512 writers, which attain marginally more than 14000 MiB/s or approximately 68% of the rated maximum, before parallel efficiency drops. As with ARCHER, this is attributed to file and disk contention.

UK-RDF DAC Performance

The UK-RDF DAC supports only shared memory parallelism; jobs cannot span multiple nodes. All tests were therefore run on a single, standard compute node offering 40 CPU cores.

We benchmarked two of the three GPFS file systems and examined the performance of each of the benchio parallel backends. Comparisons are given in Figure 16 and Figure 17.

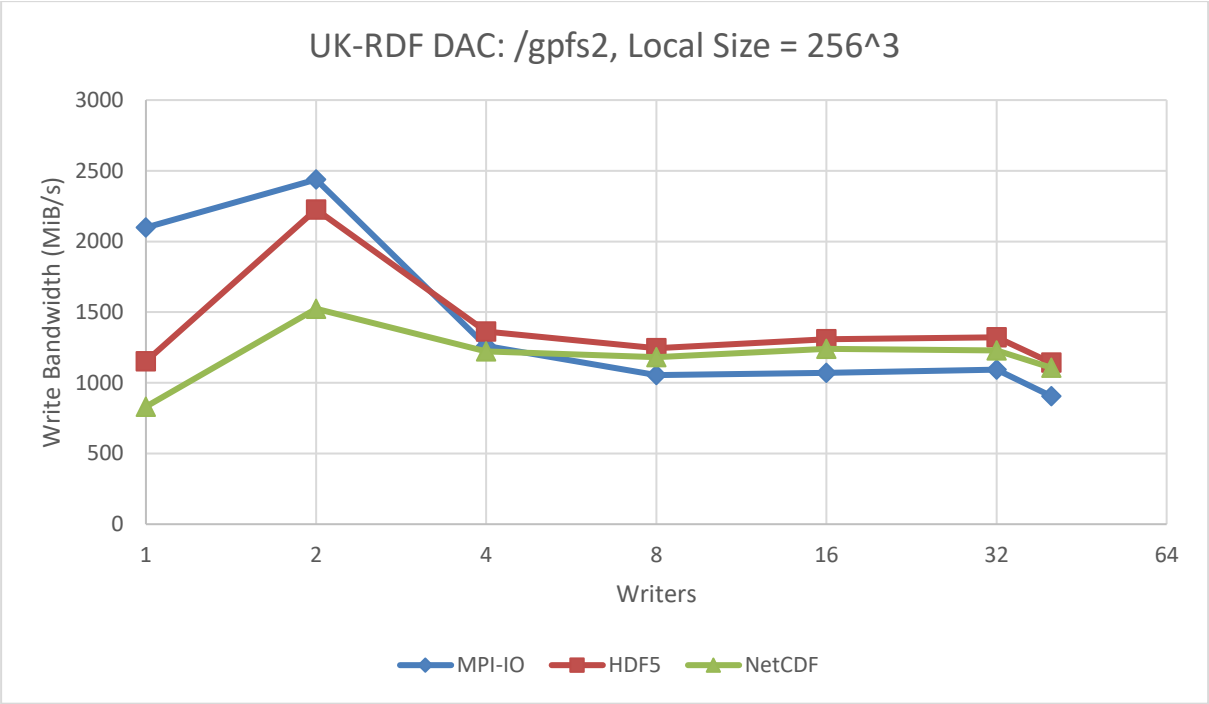

Figure 16. All backends bandwidth for UK-RDF DAC. File system: 4.4PB /gpfs2 mounted as /epsrc.

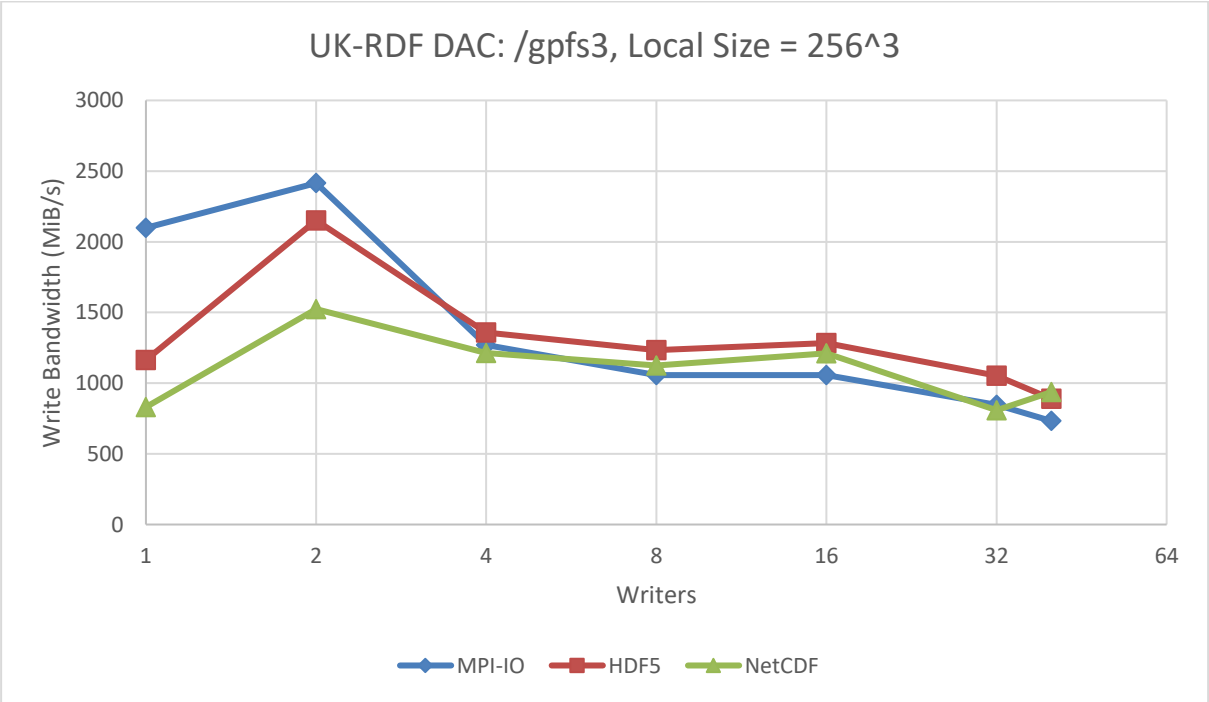

Figure 17. All backends bandwidth for UK-RDF DAC. File system: 1.5 PB /gpfs3 mounted as /general.

No difference in performance was measured between the /gpfs2 and /gpfs3 file systems. Both achieved the same peak performance of approximately 2500 MiB/s, or approximately 35% of the theoretical maximum of 7000 MiB/s. Hence file system storage capacity was found to have no bearing on overall write speed in this instance, contrary to the case of Sonexion Lustre (see the *HPC Systems* section above for an illustration of how additional storage hardware/SSUs influence the maximum potential performance of the fs4 Lustre file system on ARCHER, in comparison to fs2 and fs3).

MPI-IO, HDF5 and NetCDF displayed identical scaling characteristics with their peak bandwidths reflecting the arrangement of their hierarchy. HDF5 reached 2200 MiB/s while NetCDF performed at 1500 MiB/s, or 88% and 60% of MPI-IO respectively.

Scope for parallelisation is limited on this system with performance dropping significantly at 4 writers and above. Previous work in *Investigating Read Performance of Python and NetCDF when using HPC Parallel Filesystems*[15] on the RDF DAC supports these findings, showing sequential serial read performance to peak at roughly 1400 MiB/s, i.e. the same performance level seen from 4 to 40 writers in Figure 16 and Figure 17. Further work is needed to precisely identify the bottleneck limiting the scalability on this system.

JASMIN Performance

As with the RDF DAC, JASMIN is intended for analysis of large volumes of data. However, in contrast to the DAC, jobs can be run across multiple nodes in the cluster, potentially increasing the ceiling for parallelisation. Results were gathered from 1 to 32 writers and are presented in Figure 18.

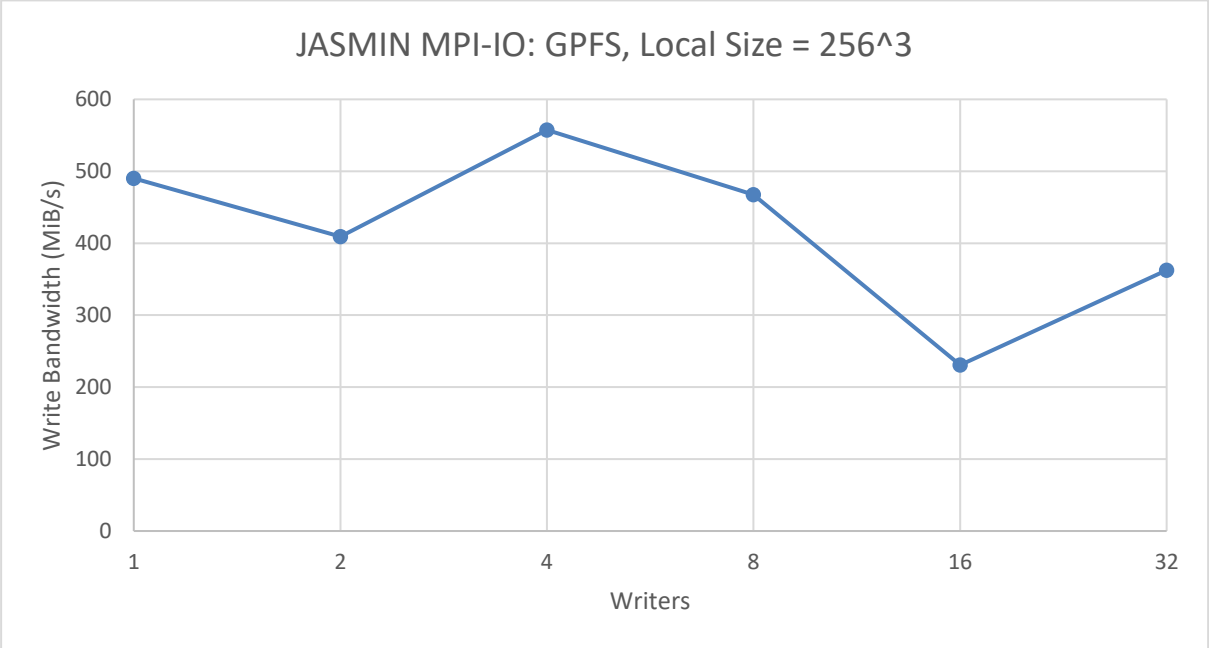

Figure 18. MPI-IO bandwidth for JASMIN

With further reference to *Investigating Read Performance of Python and NetCDF when using HPC Parallel Filesystems* [15], sequential serial performance on JASMIN has been measured at approximately 500 MiB/s, the same level of performance observed in these parallel I/O tests. From this, we conclude that there is no scope for improvement with parallelisation on this system under the default configuration. However, at time of writing, additional work is underway from Jones *et al.* to expand their investigation to include multi-threaded performance and examine parallelism on JASMIN in greater detail. Results are expected to be published at a later date.

Comparative System Performance

Figure 19 gives an overview of all four benchmark systems and compares their overall performance.

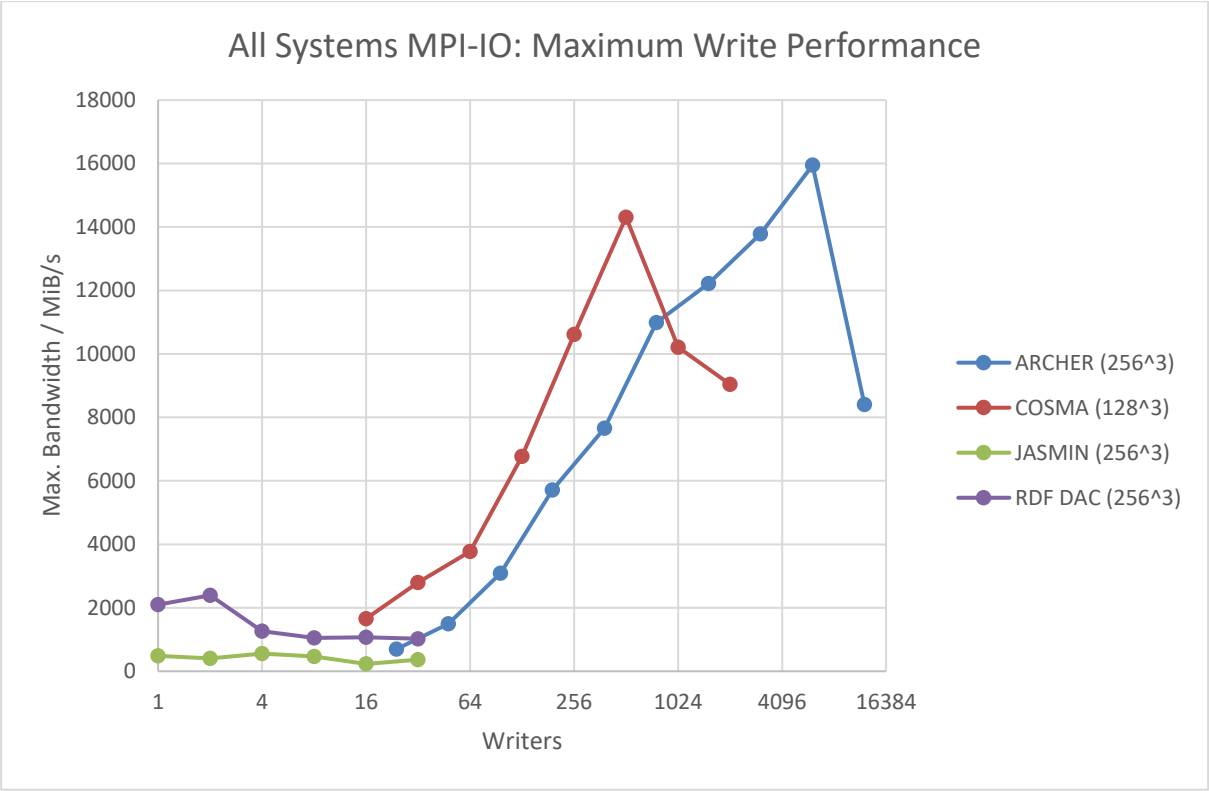

Figure 19. Comparison of maximum write performance between benchmark systems

The two systems intended for high-performance parallel simulations, ARCHER and COSMA, are broadly comparable, as are the two data analysis systems. The scope for parallelism is simply lower on JASMIN and the RDF DAC and users should not expect compute and analysis platforms to have similar performance.

Conclusions

Our findings for write performance can be summarised as follows: approximately 50% of the theoretical maximum write performance on a system should be expected to be attainable, with dramatic variance due to user contention – a factor of 200 difference in the worst case. We additionally verified that systems designed for parallel simulations offer much higher performance than data analysis platforms.

The three parallel libraries, MPI-IO, HDF5 and NetCDF, share the same performance characteristics but the higher level APIs introduce additional overhead. A reasonable expectation is 10% and 30% overhead for HDF5 and NetCDF respectively.

Tests on Lustre file systems found the optimal configuration for a single shared output file was to use maximum striping and ensure I/O operation and stripe sizes are in accordance. Generally the larger the amount of data written per writer, the larger the stripe size that should be used. Considering peak performances, improvements of approximately 10% and 35% were seen when using 4 MiB and 8 MiB stripe sizes rather than the default 1 MiB, when using large enough data sets (i.e. 256^3 array elements, or 128 MiB per writer).

Further relating to Lustre systems, users should be aware of the HDF5 performance issue and should note that versions of NetCDF below 4.4.0 should be avoided on Cray Systems as they are affected by this issue.

Finally, in contrast to Lustre, we found GPFS file system capacity to have no bearing on overall parallel I/O performance.

Future Work

Various opportunities for further investigation were identified during the course of this project. In particular, benchio could be extended to support the file-per-process I/O pattern, to complement the current work done on the single-shared-file strategy and follow-up on the bandwidth improvements in the load test shown in Figure 14. Additionally, write performance has been the exclusive focus of this work due to its relative importance in typical HPC workflows but there is scope for considering the equivalent read performance.

References

[1] NEXTGenIO | Next Generation I/O for the Exascale, http://www.nextgenio.eu/, retrieved 01 Dec 2016
[2] ARCHER HPC Resource, http://www.archer.ac.uk/, retrieved 28 Nov 2016
[3] EPCC at The University of Edinburgh | EPCC, https://www.epcc.ed.ac.uk/, retrieved 28 Nov 2016
[4] The University of Edinburgh, http://www.ed.ac.uk/, retrieved 28 Nov 2016
[5] Performance Computer, XC Series Supercomputers - Technology | Cray, http://www.cray.com/products/computing/xc-series?tab=technology, retrieved 28 Nov 2016
[6] Institute for Computational Cosmology Durham University - PhD and postgraduate research in astronomy, astrophysics and cosmology, http://icc.dur.ac.uk/index.php?content=Computing/Cosma, retrieved 28 Nov 2016
[7] DiRAC Distributed Research utilising Advanced Computing, https://www.dirac.ac.uk/, retrieved 28 Nov 2016
[8] RDF » UK Research Data Facility (UK-RDF), http://www.rdf.ac.uk/, retrieved 28 Nov 2016
[9] ARCHER » 5. UK-RDF Data Analytic Cluster (DAC), http://www.archer.ac.uk/documentation/rdf-guide/cluster.php, retrieved 28 Nov 2016
[10] home | JASMIN, http://www.jasmin.ac.uk/, retrieved 28 Nov 2016
[11] EPCCed/benchio: EPCC I/O benchmarking applications, https://github.com/EPCCed/benchio, retrieved 01 Nov 2016
[12] Jia-Ying Wu, Parallel IO Benchmarking, https://static.ph.ed.ac.uk/dissertations/hpc-msc/2015-2016/Jia-ying_Wu-MSc-dissertation-Parallel_IO_Benchmarking.pdf, retrieved 22 Nov 2016
[13] 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
[14] 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
[15] Matthew Jones, Jon Blower, Bryan Lawrence, Annette Osprey: Investigating Read Performance of Python and NetCDF When Using HPC Parallel Filesystems, http://link.springer.com/chapter/10.1007%2F978-3-319-46079-6_12, retrieved 24 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.