EUROCC - National Competence Centres in the framework of EuroHPC

EuroHPC-04-2019: HPC Competence Centres

Parallel IO on ARCHER2

Stephen Farr and David Henty
EPCC, The University of Edinburgh

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, United Kingdom, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Switzerland, Turkey, Republic of North Macedonia, Iceland, Montenegro

Table of Contents

1 Introduction

This work builds upon work previously presented by David Henty [1], which we will summarise here. ARCHER2 [2] is the UK national supercomputer, a 5860-node, 750,080 CPU-core HPE Cray EX system hosted by EPCC. It has 3 disk-based Lustre file systems, each with 12 Object Storage Targets (OSTs) and their own Meta Data Server (MDS). It also has one solid-state NVMe Lustre file system which has 20 OSTs and one MDS. All ARCHER2 users have access to one of the disk-based Lustre file systems; the NVMe file system is currently in a trial phase and only accessible by certain users.

File input and output can become bottlenecks for parallel programs running on large numbers of processors. There are three general ways to output data to disk from parallel programs:

1. Serial writing where data from all processes is sent to one controller process which writes to a single file. Having a single process writing to disk cannot take full advantage of the parallel nature of the Lustre file system, and on ARCHER2 the IO bandwidth from a single process is much less than the write speed of even a single OST.

2. File-per-process where each process writes its own local data to a separate file. This method can achieve high bandwidth in certain conditions as the files will be distributed across all the OSTs. However, Lustre filesystems are not optimised for large numbers of small files and the single MDS can become overwhelmed especially if multiple users are writing multiple files. Additionally, post processing of simulation output can become more troublesome with file-per-process output, as multiple files need to be collated and the output format depends on the number of processes used.

3. Collective parallel IO where all processes write to a single shared file in parallel. This can be done using MPI-IO or libraries such as HDF5 [3], NetCDF [4] and ADIOS2 [5]. These methods aim to overcome the limitations of the serial write, and file-per-process methods. To take advantage of a parallel filesystem requires the single shared file to be stored on multiple OSTs. For most IO libraries this requires the user to use Lustre "striping" (although ADIOS2 can exploit parallelism using a different approach).

In the previous study [1], based on a synthetic benchmark, some specific limits of IO write bandwidth on ARCHER2 were found:

- There is a per-process limit of just over 1 GiB/second.
- Per OST bandwidths are 11 GiB/s for disk and 55 GiB/s for NVMe.
- A single node can sustain up to 20 GiB/s.
- MPI-IO writing to a fully striped file (12 stripes, one per OST) on disk is limited to around 10 GiB/s. This is because MPI-IO defaults to a single IO process per stripe (an "aggregator" process) and each aggregator is limited by the per-process limit of 1 GiB/s. Using more aggregators did not improve this because it introduces added overheads from Lustre file-locking.
- Writing to NVMe using file-per-process and collective MPI-IO has similar speeds as writing to disk, despite the higher per-OST bandwidth.
- Good MPI-IO performance requires good performance from MPI collective operations. On ARCHER2, the UCX version of MPI gives much better performance than the default OFI library.
- HDF5 and NetCDF give similar performance to MPI-IO.

In this work we investigate how to increase the MPI-IO performance on ARCHER2 using the Cray Lustre lockahead [6] MPI-IO runtime options which were not previously investigated. We then investigate the performance of ADIOS2 [5], a more modern parallel IO library, and look at the performance of IO from a real HPC application rather than a synthetic benchmark,

2 Methodology

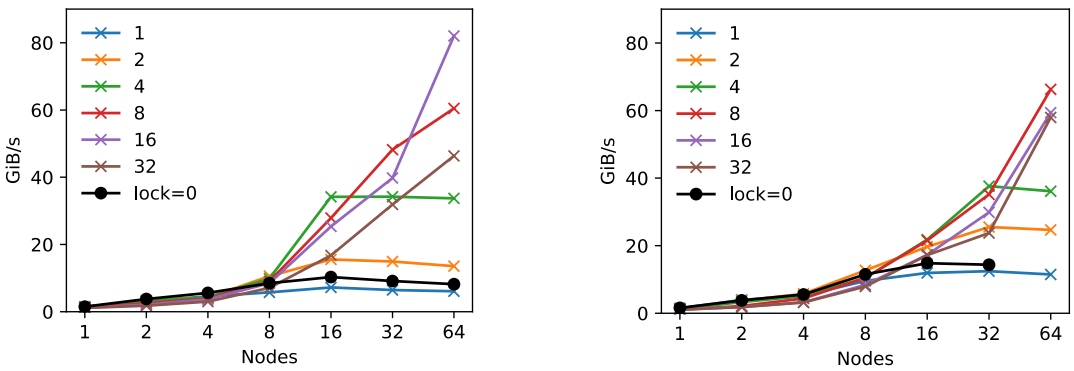
Unless otherwise stated all performance results were produced using the programming environment PrgEnv-gnu/8.0.0. As we are interested in the maximum obtainable IO bandwidth, we repeat measurements 10 times and take the maximum results. We ensure to only run one IO benchmarking job at once so we are not clashing with our own application. MPI-IO and HDF5 results use fully-striped files, set using the Lustre command `lfs setstripe -c -1`. This equates to 12 stipes on the disk filesystem and 20 stripes on NVMe. The stripe size is left at the default of 1MiB. For ADIOS2 unstriped files are used because, for the native BP5 file format used here, parallelism is achieved by writing a large number of files so does not rely on Lustre file striping. Raw data is available in "results" at https://github.com/davidhenty/benchio/.

2.1 Lustre lockahead

When writing to shared files on Lustre the IO performance is limited by Lustre Locking mechanisms when using more than one aggregator process per stripe. These overheads can be overcome using the Cray lockahead feature [6]. Lockahead is enabled at runtime using the MPIIO_HINTS environmental variable

```
export MPICH_MPIIO_HINTS=*:cray_cb_write_lock_mode=2,*:cray_cb_nodes_multiplier=N
```

where N is an integer. The "cray_cb_write_lock_mode=2" setting turns on Lustre lockahead. A value of 0 is default locking and a value of 1 is group locking which was found to have no performance benefit on ARCHER2 [1]. The "cray_cb_node_multipler=N" sets the number of aggregators per stripe; "*" applies these settings to all files.



A. Disk **B. NVMe**

Figure 1 ***A****. Disk strong scaling performance of benchio using MPI-IO.* ***B****. NVMe strong scaling performance of benchio using MPI-IO. The legend gives the number of aggregators per stripe with lustre lockahead enabled. The black circle data points are for default Lustre locking (lock=0).*

Using a synthetic IO benchmarking program *benchio* [7] we investigated the strong scaling performance of MPI-IO using Lustre lockahead with different numbers of aggregators per stripe on disk and NVMe. The benchmarks were done using a global array size of 64GiB writing to a fully striped file (12 stripes for disk, 20 stripes for NVMe) with a stripe size of 1MiB using the UCX library. The results are plotted in figure 1 for [1, 2, 4, 8, 16, 32] aggregators per stripe, the performance of lockmode=0 (default locking mode, 1 aggregator per stripe) is also plotted for comparison. Subfigure A shows the results for disk and subfigure B those for NVMe.

We see that for both disk and NVMe that for up to 8 nodes there is not much difference between the default lockmode=0 and the different aggregator settings. In fact lockmode=0 gives the best performance for 4 nodes or less. In this regime the overheads of Lustre lockahead appear greater than any performance benefit. For disk we see a significant difference at 16 nodes: using 2 or more aggregators gives increased performance, with 4 aggregators per stripe giving the best performance of 35 GiB/s which is a more than 3x increase over the default locking. The performance difference between 8 and 16 nodes is approximately 4x which is greater than the linear scaling increase of 2x. This could be attributed to the per-node bandwidth discussed previously: with 8 nodes there are more OSTs than nodes, while with 16 Nodes there are more nodes than OSTs which helps to overcome the per-node bandwidth bottleneck.

Interestingly for disk we see that setting the aggregators per stripe to the number of nodes/4 gives maximum performance. The peak bandwidth is 80 GiB/s for 64 nodes with 16 aggregators per stripe which approaches the quoted hardware maximum of 131 = 12 OST * 11 GiB/s. For NVMe we see less of a difference between the aggregator settings with 8 aggregators per node giving good performance for all node counts. For 16 or more nodes the maximum bandwidth is lower than disk. This is unexpected and not readily explainable.

2.2 Lockahead with HDF5

We then checked how the lockahead settings affect HDF5 output. To do this we ran *benchio* using the HDF5 output setting for 64 nodes with different aggregators per stripe and lockahead enabled. The results are plotted in figure 2 with the MPI-IO results for comparison. We see that HDF5 performance closely follows MPI-IO while mostly being slightly lower. The peak performance is still reached with 16 aggregators per stripe. This is as expected because HDF5 runs on top of MPI-IO and the additional overheads account for the slight drop in performance.

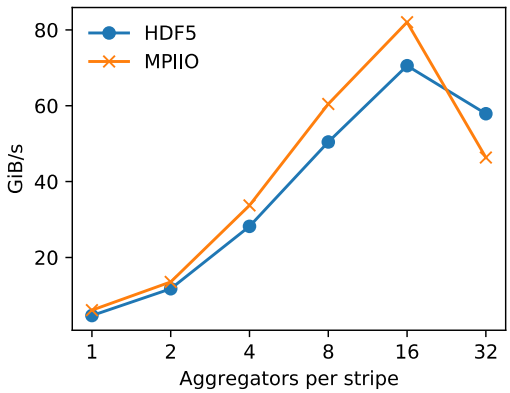


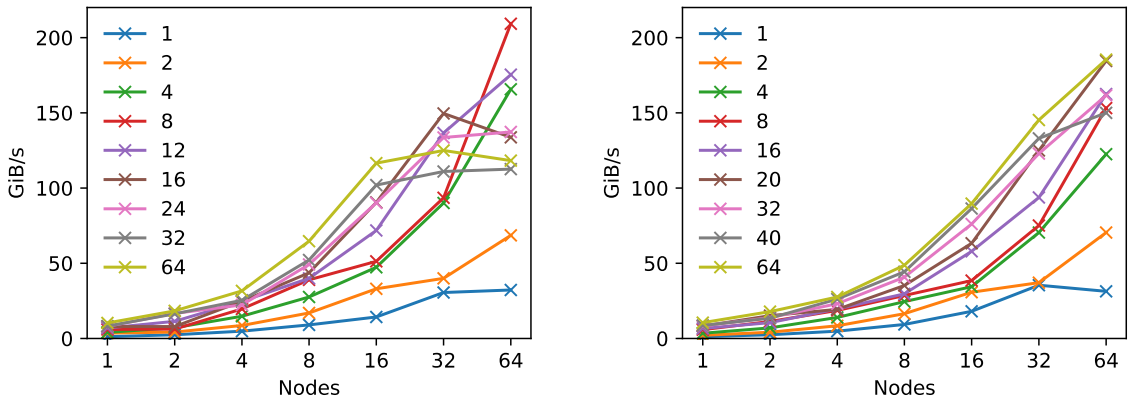
Figure 2. HDF5 benchio performance compared with MPI-IO for 64 nodes using different aggregators per stripe with lockahead enabled.

2.3 ADIOS2

As part of this new study we implemented ADIOS2 (v2.8.3) as a new output option in benchio. ADIOS2 is a parallel IO library that can write files in a variety of formats including HDF5. In this work we focus on the BP5 file format, the most recent version of the ADIOS2 binary-pack native format. A BP5 file is actually a directory; within the directory are the data files which contain the binary data (1 data file is written per aggregator) and the metadata files. The number of aggregators is the parameter than can be tuned to optimise ADIOS2 performance. The default value is 1 aggregator per shared memory node, but it can be controlled at runtime by providing the desired number in the ADIOS2 config file. An example is shown below for 4 aggregators.

```
“adios2_config.xml”
<?xml version="1.0"?>
<adios-config>
  <io name="Output">
    <engine type="BP5">
      <parameter key="NumAggregators" value="4"/>
    </engine>
  </io>
</adios-config>
```

We investigated the strong scaling performance for the same 64GB file size as figure 1. The number of aggregators per node was varied from 1 to 64. The results are plotted in figure 3.



A. Disk **B. NMVe**

Figure 3 ***A****. Disk strong scaling performance for benchio using ADIOS2.* ***B****. NVMe strong scaling performance for benchio using ADIOS2. The legend gives the number of aggregators per node.*

We see that for 16 or fewer nodes the performance is increased when more aggregators are used, e.g. the maximum for 16 nodes is at 64 aggregators per node. For large node counts on disk the maximum performance is given by fewer aggregators with peak performance of 200 GiB/s achieved using 8 aggregators per node with 64 nodes. The performance of ADIOS2 is more than double MPI-IO. This is because the BP5 output format uses multiple files, therefore it is affected by file locking and scales similarly to the file-per-process output presented in [1]. Importantly it can achieve good performance using 10x fewer files than file-per-process, thus putting less strain on the MDS, and by writing into a shared file (actually a directory) aids post processing and portability. The fact that the IO performance for high node counts exceeds the peak performance of the disk filesystem indicates that some file caching may be occurring.

3 Applying findings to a full HPC application

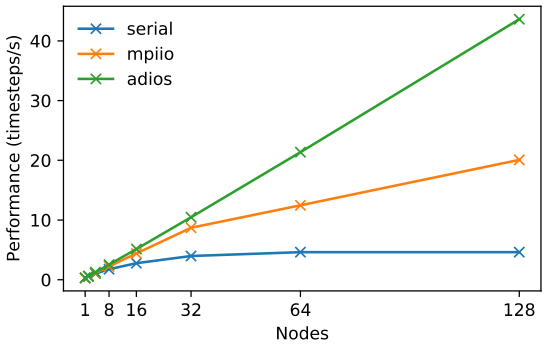


Figure 4. Strong scaling performance of LAMMPS for 500 million atom system with different IO options. Serial uses the default "dump atom" settings. MPI-IO uses "dump atom/mpiio" with lockmode=2, and the aggregators per stripe set to nodes/4. ADIOS uses "dump atom/adios" with ADIOS2 BP5 file format with 8 aggregators per node.

To verify if the findings from our synthetic benchmark are applicable to general HPC applications we investigate the IO performance of LAMMPS [8] which is a molecular dynamics code written in C++ using MPI. The program architecture, MPI communications and file output formats are different to the benchio code. For a test system we used a 500 million atom Lennard-Jones atomic fluid, created by scaling up the LJ benchmark system provided by LAMMPS. The potential is a Lennard-Jones interaction with a 2.5 sigma cutoff; there are no long range interactions so the parallel scaling is expected to be good. The standard benchmark does not include any IO, however, for such a large system (20 GB coordinate snapshot file) the IO can become a bottleneck. We modified the benchmark to run for 100 timesteps and then write a snapshot of the atomic coordinates to file. We did this using the default LAMMPS dump command "dump atom" which uses a serial method where all ranks send local data to rank zero and rank zero writes the output. We also used an MPI-IO method "dump atom/mpiio" and an ADIOS2 method "dump atom/adios" to output the same data but using collective parallel IO. For the MPI-IO case we used the optimal settings found in the previous section: lockahead was turned on and the aggregators per stripe were set to nodes/4. For ADIOS2 case we used BP5 format and used 8 aggregators per node. We ran with 1, 2, 4, 8, 16, 32, 64 and 128 nodes and measured the performance in timesteps per second (simulation timesteps per second of wall time), where this timing incudes the time taken to run 100 timesteps and write the atomic coordinates. The results are plotted in figure 4. We see that with the serial output method the scaling is very poor, plateauing at 16 nodes For MPI-IO the situation is improved, but even on 32 nodes starts to deviate from ideal linear scaling. For the ADIOS2 output method we see ideal linear scaling. These results demonstrate the same 2x performance increase of ADIOS2 over MPI-IO that we observed in the previous section, and that with lockahead enabled MPI-IO is able to scale with increasing node count.

4 Conclusions

We found that enabling Lustre lockahead and using multiple aggregators per stripe greatly increases MPI-IO output rates to a shared file on ARCHER2.Without these settings the IO is capped at a low value of around 10GiB/s; using them we can achieve 80 GiB/s. The difficulty in achieving high parallel IO rates on ARCHER2 is heavily controlled by the 1GiB/s per-process limit and the per-node bandwidth: it is difficult to saturate the OST bandwidths without

using a large number of nodes. ADIOS2 IO offers significant performance gains, with minimal user tuning needed, because its native BP5 output format writes multiple files thus overcoming the per-process IO limits. While the collective IO performance on the Lustre disks can reach the hardware maximum, the performance on the NVMe partition is currently poor, offering no benefit and in some cases being worse than disk. Further work is needed to investigate how the NVMe storage can be better utilised.

4.1 General guidance for ARCHER2 users

When using MPI-IO, or MPI-IO-based libraries such as HDF5, for parallel IO it is important to tune the MPI-IO settings when using more than 8 nodes. Users should switch to the UCX library, turn on Lustre lockahead and set the number of aggregators per stripe to nodes/4. The command to do this on ARCHER2 at runtime in a slurm script are:

```
module swap craype-network-ofi craype-network-ucx
module swap cray-mpich cray-mpich-ucx

export MPICH_MPIIO_HINTS=*:cray_cb_write_lock_mode=2,*:cray_cb_nodes_multiplier=N
```

where N should be set to an integer equal to the number of nodes divided by 4.

Furthermore, if available, users should consider using ADIOS2 with the BP5 file format and setting the number of aggregators per node to 8.

5 References

[1] D. Henty, “Performance of Parallel IO on the 5860-node HPE Cray EX System ARCHER2”, presented at CUG2022, The Cray User Group, Monterey, CA, 2-5 May 2022.
[2] https://www.archer2.ac.uk/
[3] M. Folk et al., “An overview of the HDF5 technology suite and its applications”, Proceedings of the 2011 EDBT/ICDT Workshop on Array Databases, March 25 2011, http://dx.doi.org/10.1145/1966895.1966900
[4] R. Rew and G. Davis, “NetCDF: an interface for scientific data access” in IEEE Computer Graphics and Applications, vol. 10, no. 4, pp. 76-82, July 1990, doi: 10.1109/38.56302.
[5] W. F. Godoy et al.,, “ADIOS 2: The Adaptable Input Output System. A framework for high-performance data management”, SoftwareX, Volume 12, 2020 100561, ISSN 2352-7110, https://doi.org/10.1016/j.softx.2020.100561.
[6] M. Moore, P. Farrell and B. Cernohous, “Lustre Lockahead: Early experience and performance using optimized locking”, *Concurrency Computat: Pract Exper*. 2018; 30:e4332. DOI:10.1002/cpe.4332
[7] https://github.com/davidhenty/benchio
[8] A. P. Thompson et al., LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, *Comp Phys Comm*, 271 (2022) 10817. DOI:10.1016/j.cpc.2021.108171