THE UNIVERSITY of EDINBURGH

epcc

Benchmarking the I/O performance of the world's 5th largest CPU-based Supercomputer, ARCHER2

Shrey Bhardwaj, Dr. Paul Bartholomew, Prof. Mark Parsons

EPCC, University of Edinburgh

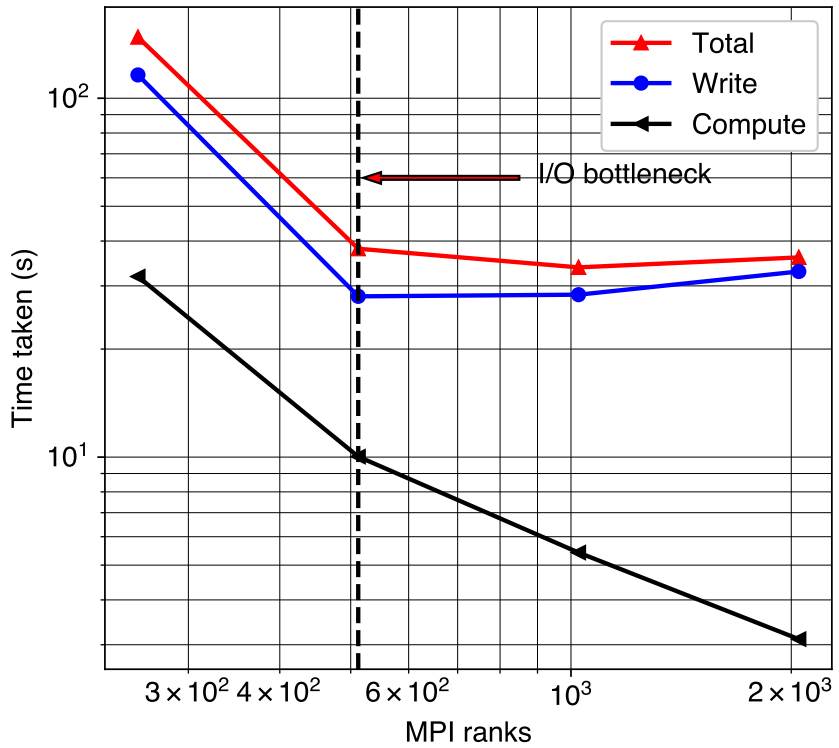
ISC High Performance 2022

ASIMOV

archer2

Outline

- Addressing the I/O bottleneck is key for exascale computing.



XCompact3D benchmarking [1]

[1] Fulhame dataset `https://github.com/pbartholomew08/X3D-benchmarking.git`

Novel methods in I/O

- C based benchmark_c[1] derived from Fortran based benchio[2]
- Performs I/O benchmarking in different I/O backends; MPIIO, HDF5, ADIOS2
- Added features in bechmark_c include:
  - + ADIOS2 layer with runtime config file to use different backends
  - + Can read external datasets like CRESCENDO BDA datasets.
- ADIOS2[3]:
  - + Open source framework for scientific parallel I/O
  - + Low level support
  - + High level APIs and command line tools
  - + Abstractions provided for different I/O layers; HDF5,BP4, BP5
  - + Can initialise with config file to reduce need for re-compilation

[1] `https://github.com/sb15895/benchmark_c.git`

[2] `https://github.com/davidhenty/benchio.git`

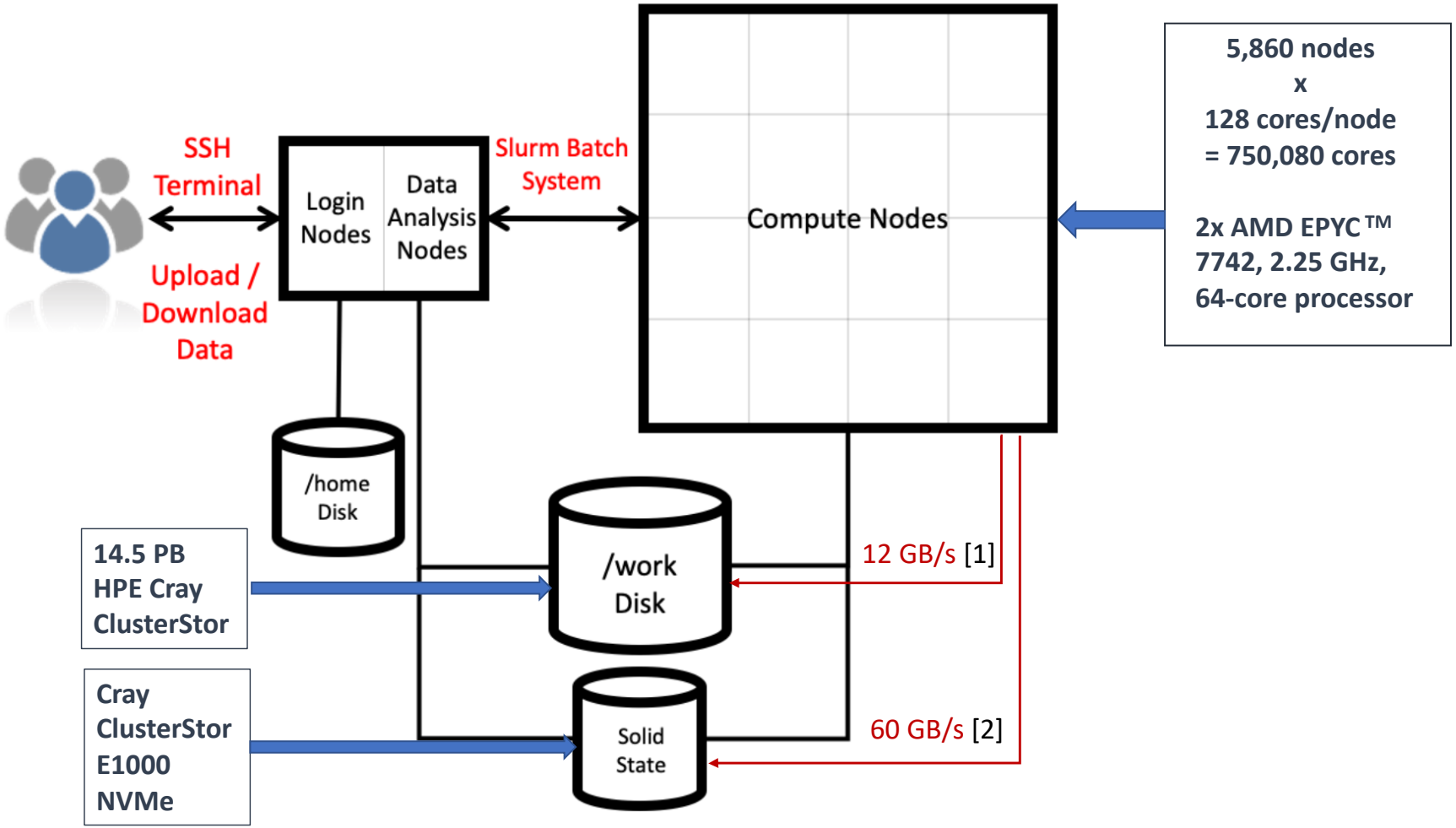
[3] `https://github.com/ornladios/ADIOS2.git`

- Next generation UK National Supercomputing Service
- Provided by UKRI, EPCC, Cray (an HPE company) and the University of Edinburgh[1].
- Estimated peak performance of 28 Pflop/s
- Average speedup compared to predecessor ARCHER were the following[2]:
  - 8.7x for CP2K
  - 9.5x for OpenSBLI
  - 11.3x for CASTEP
  - 12.9x for GROMACS
  - 18.0x for the HadGEM3 climate model
- Achieved 10th Position in IO500 in SC21 list,IO500 score of 10.68[3]

[1] `https://docs.archer2.ac.uk`

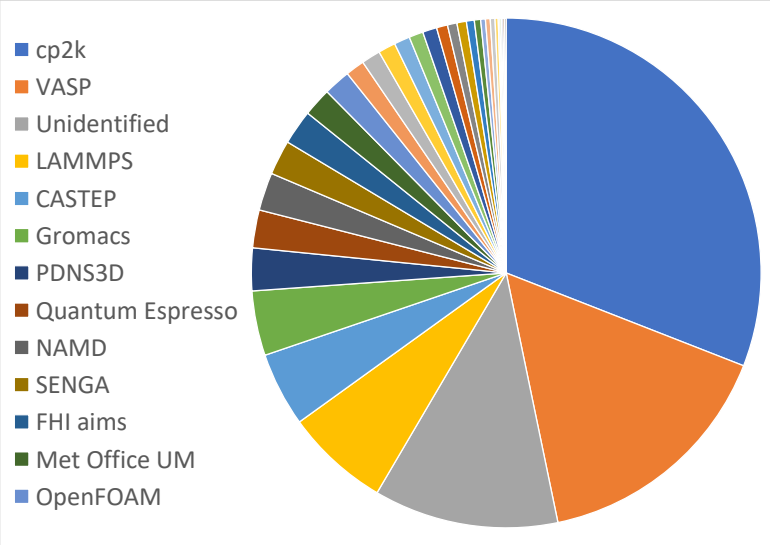
[2] `https://www.archer2.ac.uk/about/hardware.html`

[3] `https://io500.org/submissions/view/472`

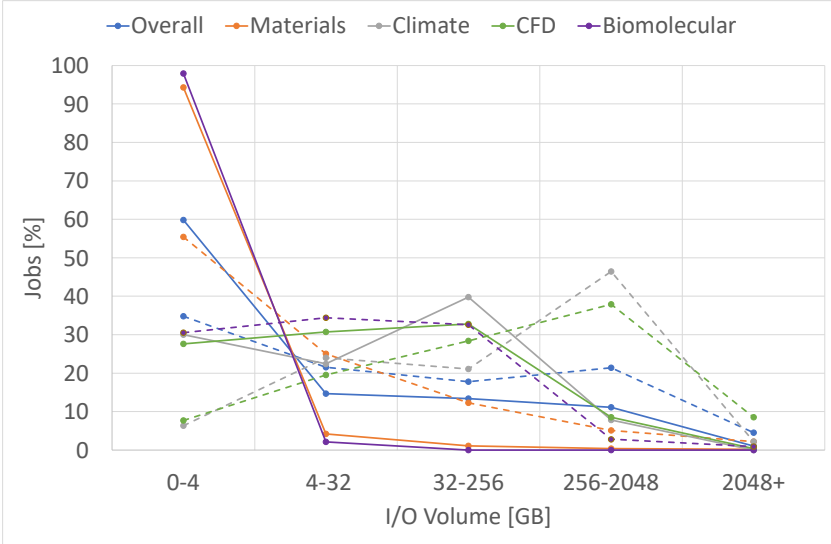


[1] https://docs.archer2.ac.uk/user-guide/hardware/

[2] https://www.hpe.com/psnow/doc/a00062172enw.pdf?jumpid=in_pdp-psnow-qs



ARCHER2 usage by CU [1]

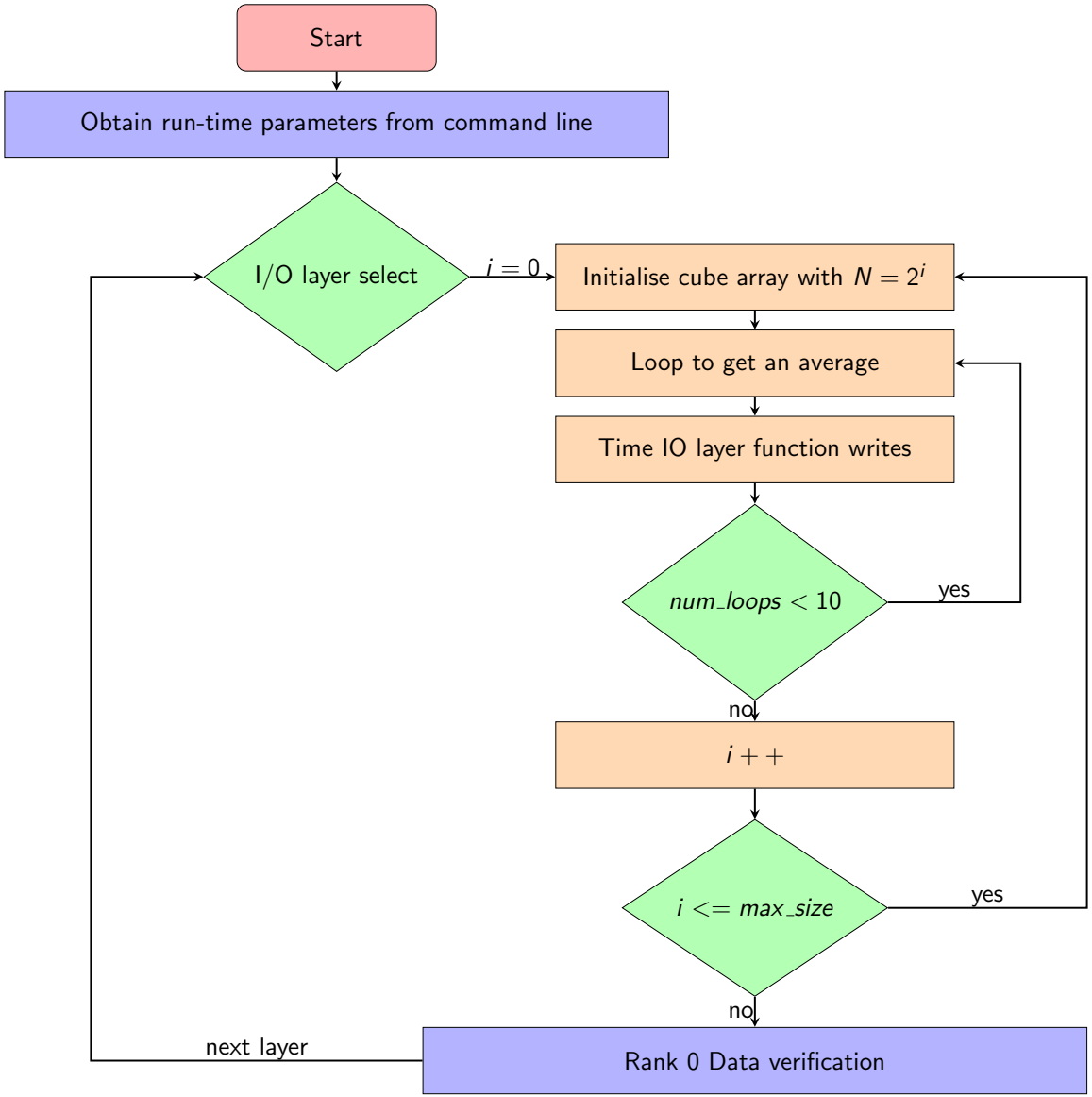


ARCHER I/O usage statistics [2]

[1] https://www.archer2.ac.uk/news/2021/05/19/code-use.html

[2] https://cug.org/proceedings/cug2019_proceedings/includes/files/pap118s2-file1.pdf

Outline


Start
Obtain run-time parameters from command line
I/O layer select
$i = 0$
Initialise cube array with $N = 2^i$
Loop to get an average
Time IO layer function writes
$num\_loops < 10$
yes
no
$i++$
$i <= max\_size$
yes
no
next layer
Rank 0 Data verification

```
// MPIIO write function
mpiiowrite(iodata, local_size, global_size, arraystart, NDIM, cartcomm,
           output_file[iolayer]);
// PHDF5 write function
phdf5write(iodata, local_size, global_size, arraystart, NDIM, cartcomm,
           output_file[iolayer]);
// ADIOS2 write function HDF5
adioswrite(iodata, local_size, global_size, arraystart, NDIM, cartcomm,
           "HDF5", output_file[iolayer]);
// ADIOS2 write function BP4
adioswrite(iodata, local_size, global_size, arraystart, NDIM, cartcomm,
           "BP4", output_file[iolayer]);
// ADIOS2 write function BP5
adioswrite(iodata, local_size, global_size, arraystart, NDIM, cartcomm,
           "BP5", output_file[iolayer]);
```

ADIOS2 runtime configuration

ADIOS2 runtime configuration

Variable passed IO_engine="BP4"

```
adios2_adios *adios = adios2_init_config(config_file, cartcomm, 1);
adios2_io *io = adios2_declare_io(adios, IO_ENGINE);
adios2_variable *var_iodata = adios2_define_variable(io, "iodata",
    adios2_type_double, NDIM, shape, start, count, adios2_constant_dims_true);
adios2_engine *engine = adios2_open(io, FILENAME, adios2_mode_write);
adios2_step_status status;
adios2_begin_step(engine, adios2_step_mode_update, 10.0, &status);
adios2_put(engine, var_iodata, iodata, adios2_mode_deferred);
adios2_end_step(engine);
adios2_close(engine);
adios2_finalize(adios);
```

ADIOS2 config file initialises BP4 engine

```
<?xml version="1.0"?>
<adios-config>
  <io name="BP4">
    <engine type="BP4">
    </engine>
  </io>
  <io name="HDF5">
    <engine type="HDF5">
    </engine>
  </io>
  <io name="BP5">
    <engine type="BP5">
    </engine>
  </io>
</adios-config>
```

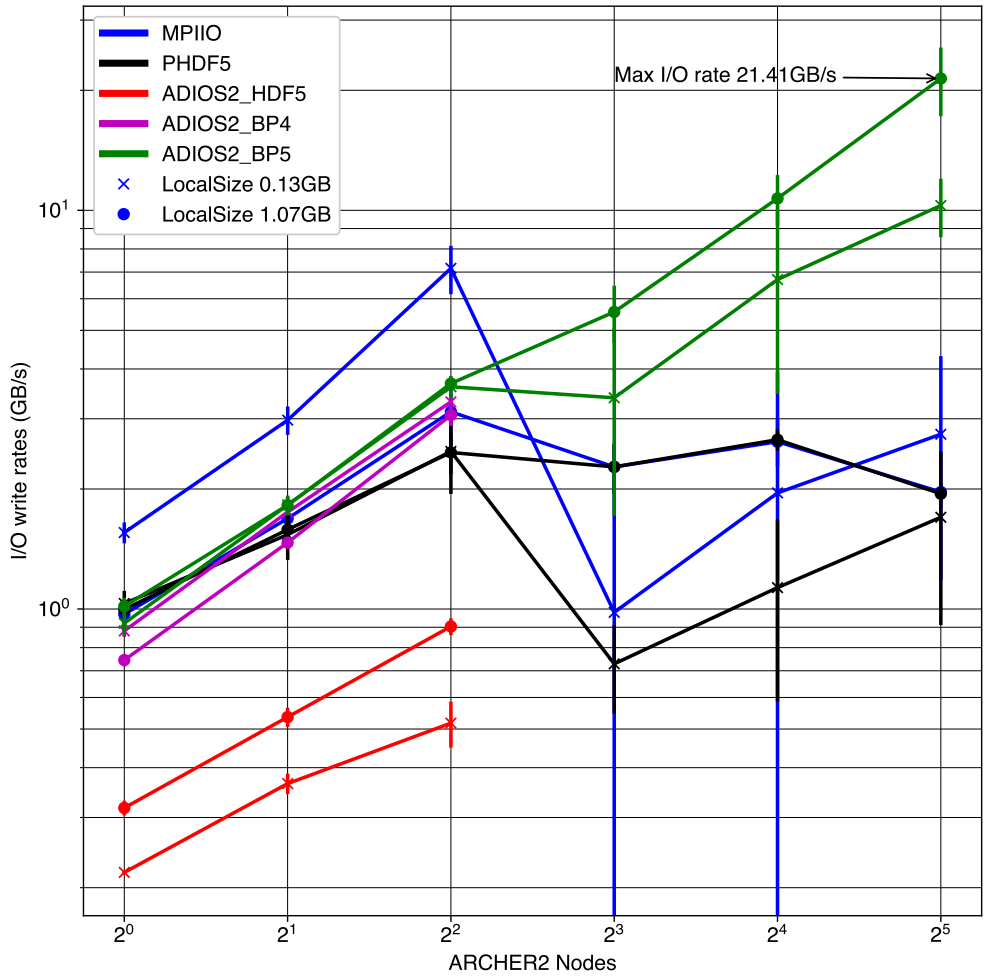
- Modules and libraries loaded

I/O library	Version
Prg-Env-gnu	8.0.0
craype	2.7.6
GCC	10.2.0
Cray MPICH	8.1.4
Cray HDF5 parallel	1.12.0
ADIOS2	2.8.0
DARSHAN/Runtime	3.3.1

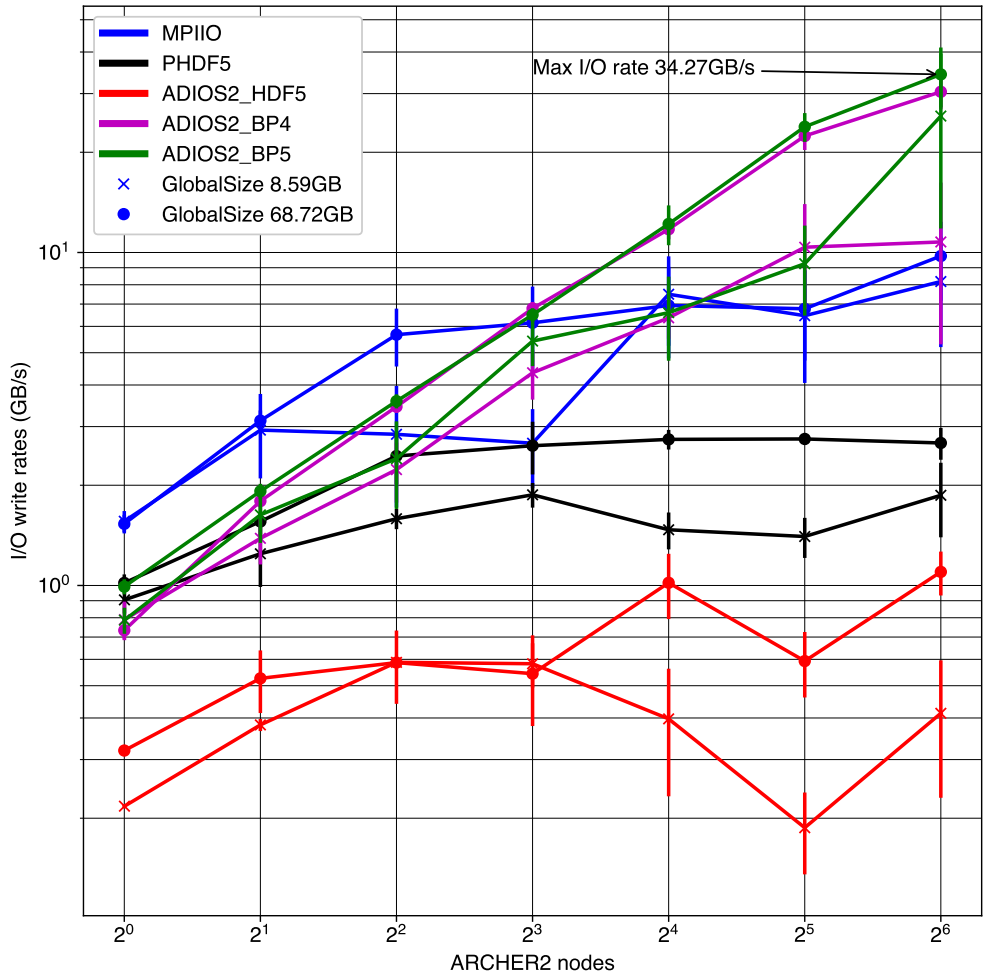
- Environment flags used; FI_OFI_RXM_SAR_LIMIT=64K [1]

[1] https://docs.archer2.ac.uk/user-guide/io/

Outline

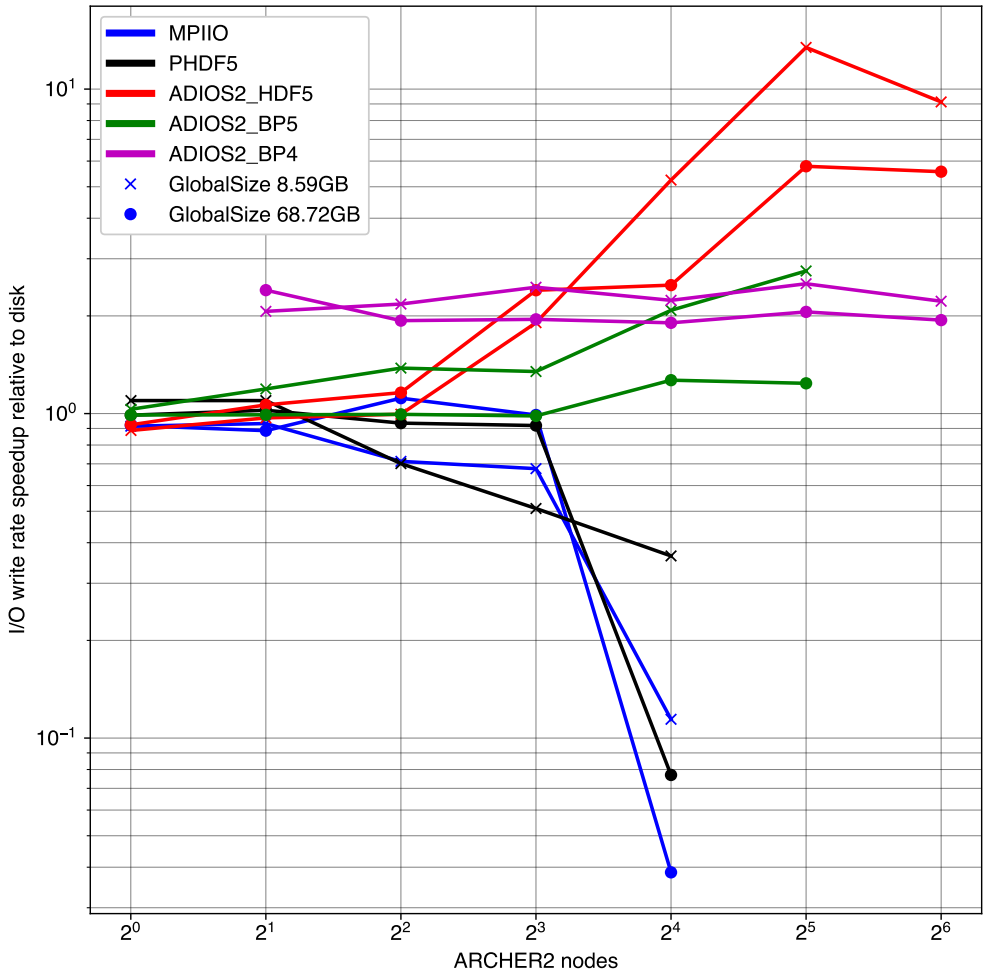


ARCHER2 weak scaling results upto local size 1.07GB

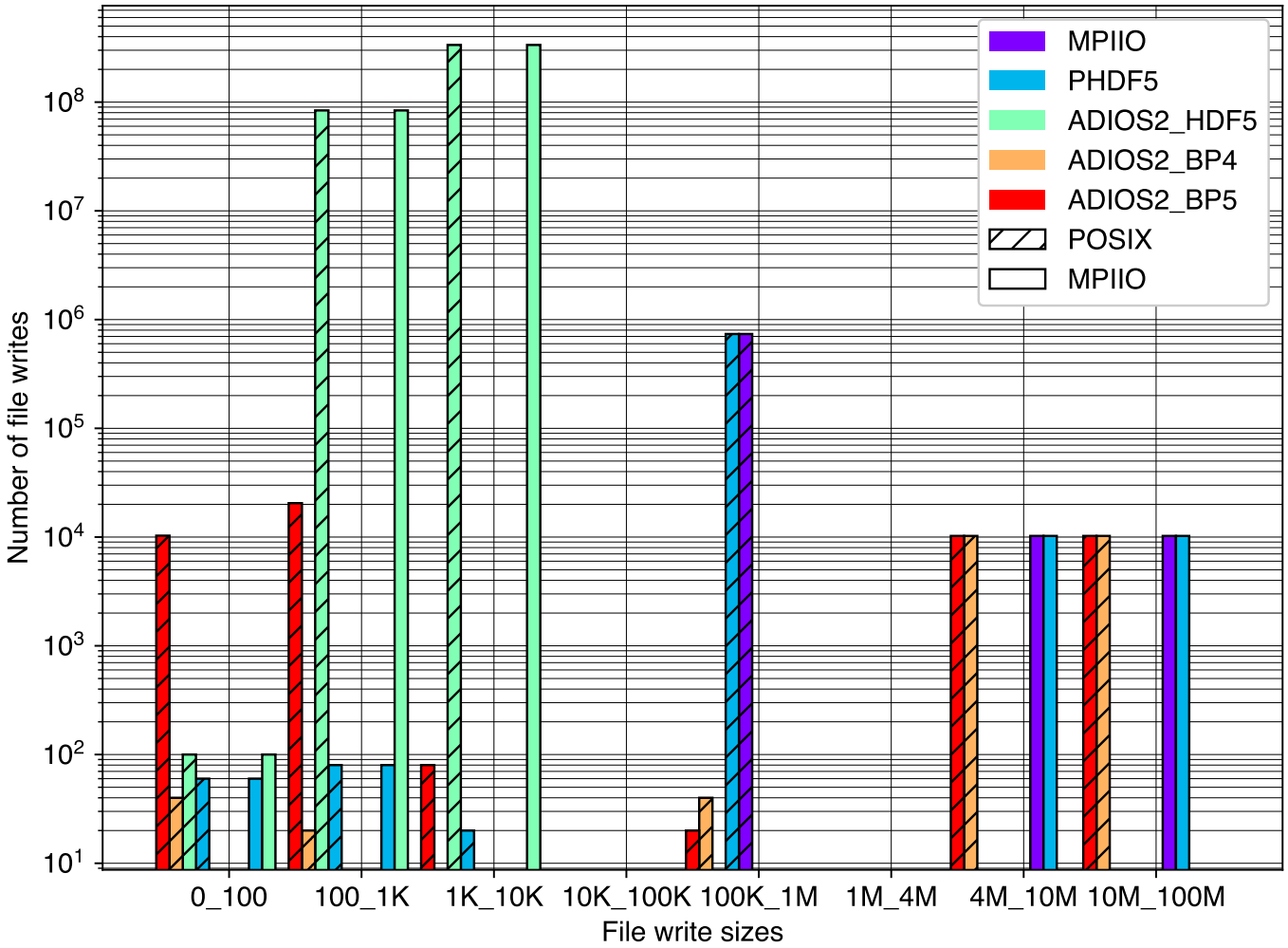


ARCHER2 strong scaling results upto global size 68.72GB

NVMe layer results speedup



ARCHER2 NVMe strong scaling speedup upto global size 68.72GB



DARSHAN I/O filesize distribution using 8 full nodes, 64GB global size

Conclusions

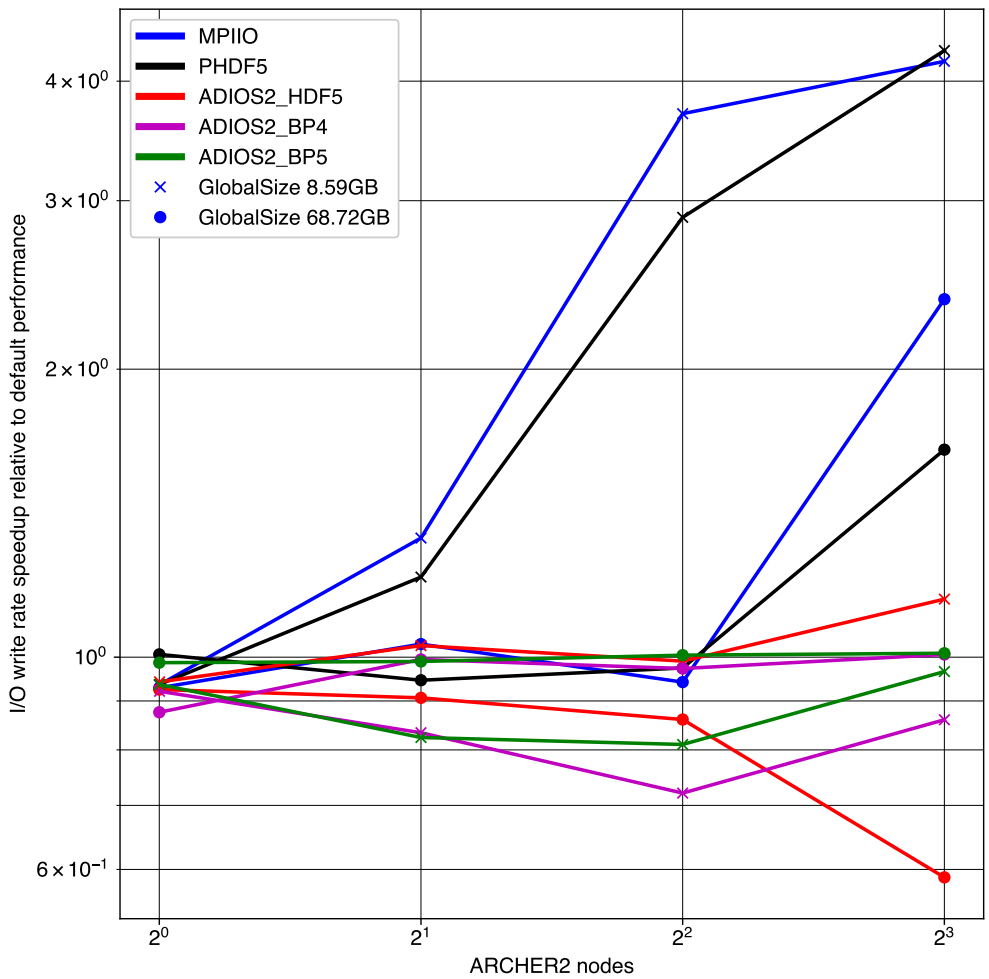
- ADIOS2 BP4 & BP5 scale more effectively v/s MPI, HDF5.
  - + DARSHAN profiler shows BP4 & BP5 layers create fewer file writes
  - + ADIOS2 BP4 & BP5 uses writers/node
  - + Improved performance can be due to less contention when writing
- ADIOS2 HDF5 performs slower than HDF5 as per results
  - + DARSHAN profiler shows larger no. of files generated v/s HDF5.
  - + The reason is yet to be explored.
- NVMe layer on ARCHER2 shows speedup in some layers
  - + Except MPI, PHDF5 which have negative performance
  - + Highest write rate achieved was 44GB/s from ADIOS2_BP4
  - + NVMe layer is not yet fully established, but needs more analysis.

Links

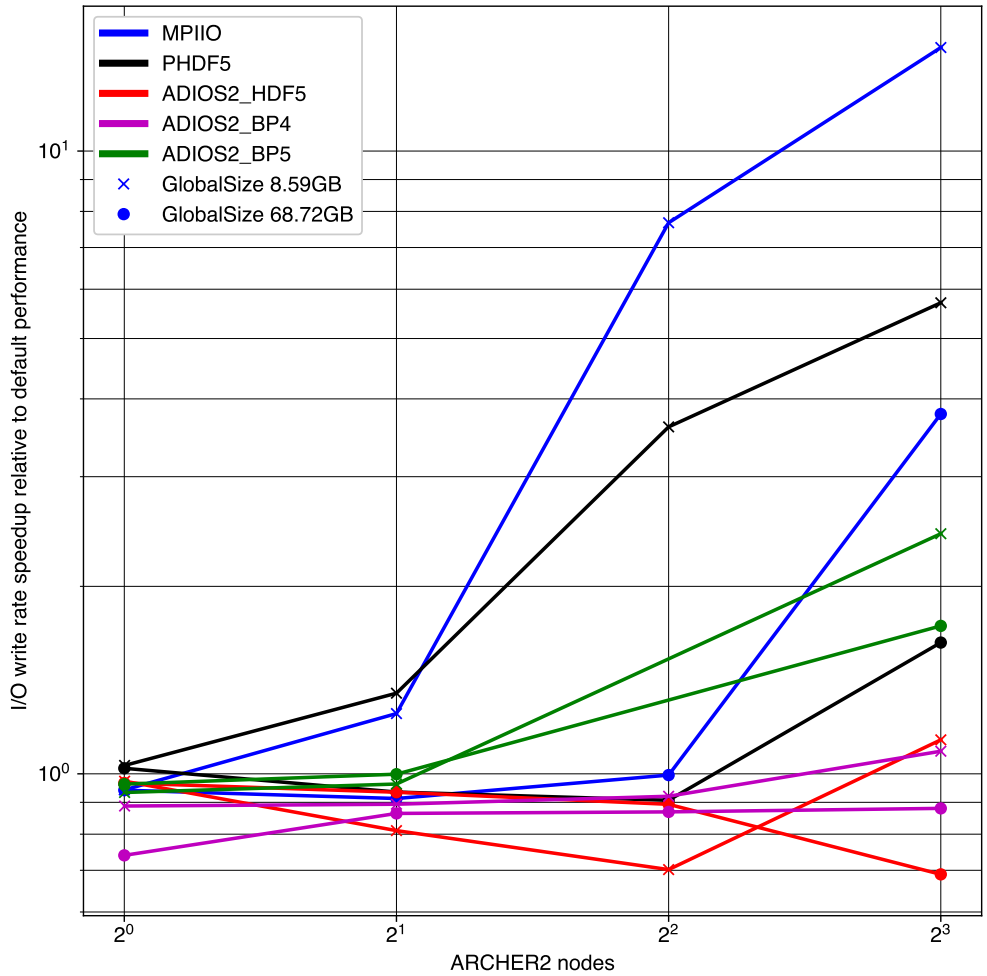
→ email: shrey.bhardwaj@epcc.ed.ac.uk

→ github: `https://github.com/sb15895/benchmark_c.git`

Outline



Impact on I/O performance from changing SAR LIM flag 8k to 64k



Impact on I/O performance from using non-default UCX MPI

Calls comparison