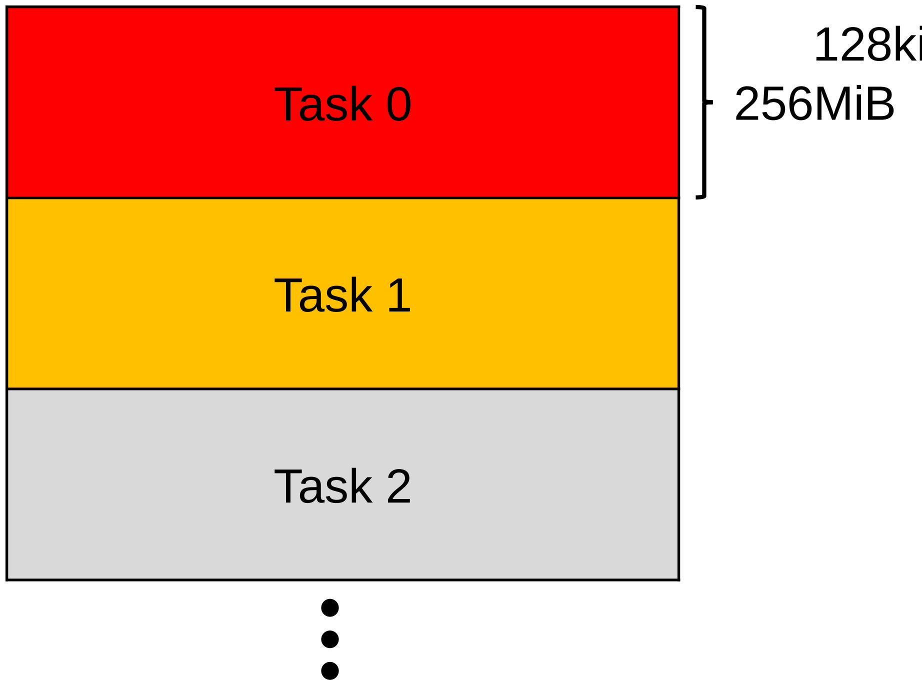
PARALLEL I/O AND PORTABLE DATA FORMATS

OPTIMIZATION AND PROFILING

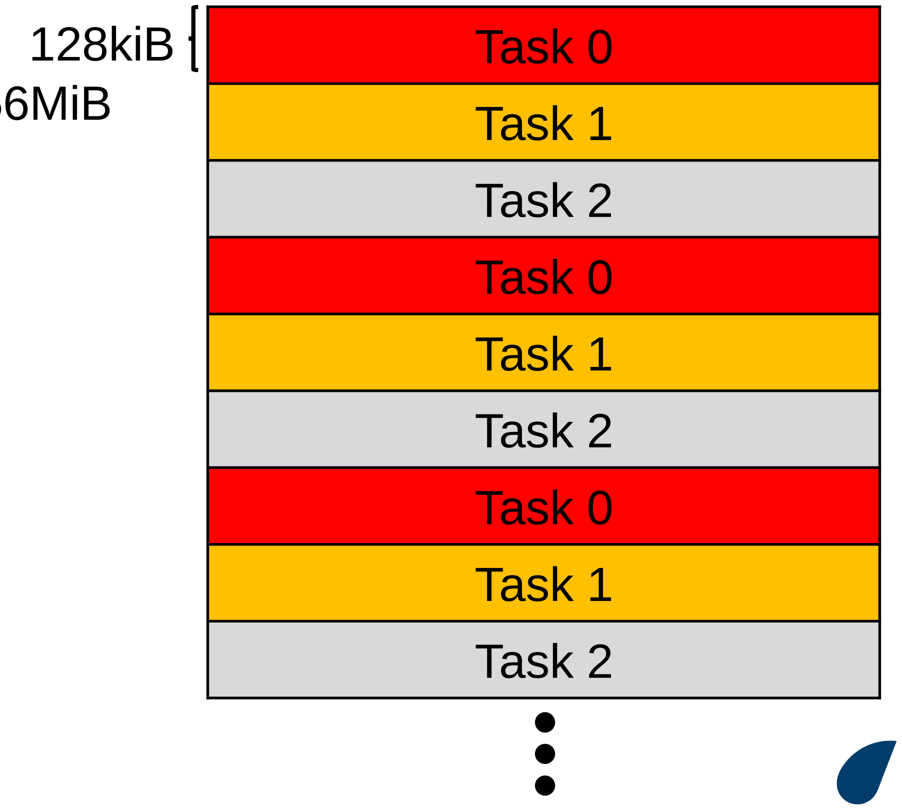
23.02.2022 I SEBASTIAN LÜHRS (S.LUEHRS@FZ-JUELICH.DE)

I/O patterns

continuous
- Large continuous data blocks for each individual process



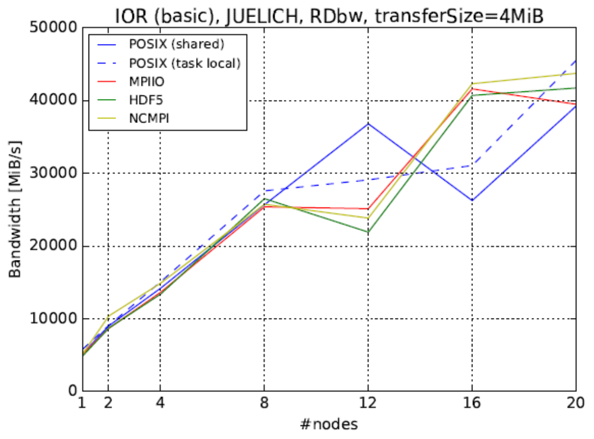
striped
- Pattern often found while handling multi dimensional arrays

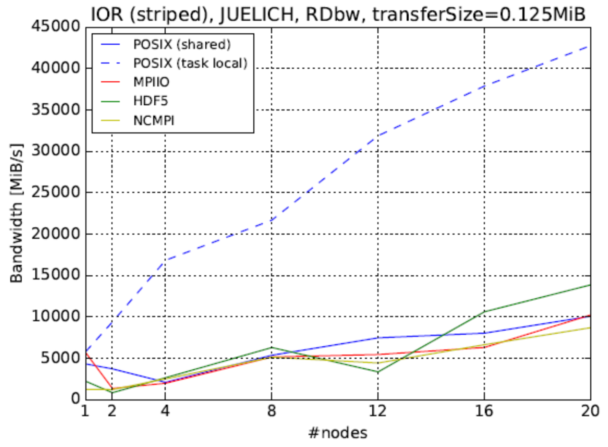


I/O pattern bandwidth

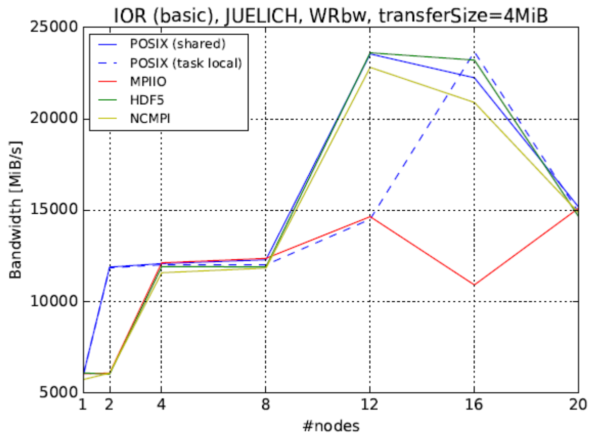
continuous **striped**

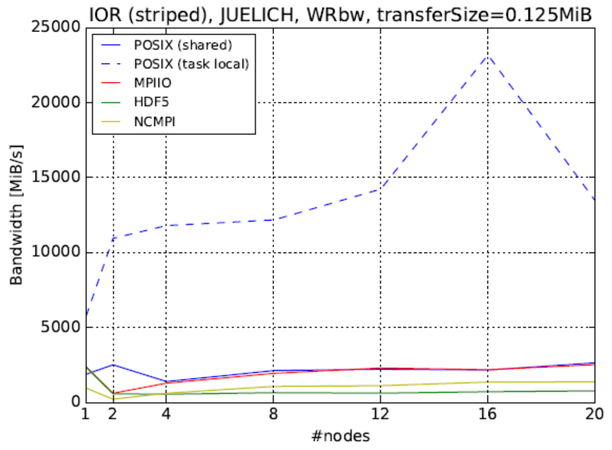
read bandwidth





write bandwidth





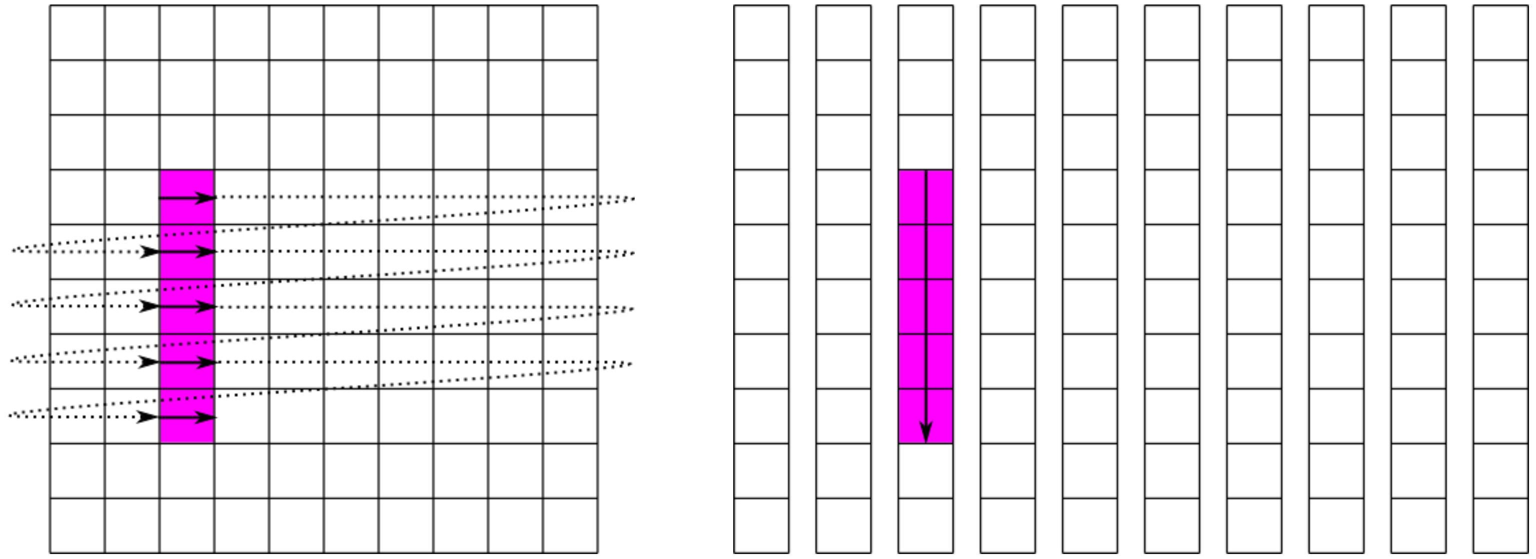
Measurements on JURECA at JSC

This work was supported by the Energy oriented Centre of Excellence (EoCoE), grant agreement number 676629, funded within the Horizon2020 framework of the European Union.

Performance hints

Chunking

- Contiguous datasets are stored in a single block in the file, chunked datasets are split into multiple chunks which are all stored separately in the file.
- Additional chunk cache is possible



https://www.hdfgroup.org/HDF5/doc/Advanced/Chunking/

Performance hints

Compression

- In-transit compression can help to lower the overall datasize:
- HDF5 (and NetCDF4) allows compression within a parallel, collective write commands for chunked datasets
- Gzip (`deflate`) compression available by default (szip can be added on demand)
- Other compression techniques are available by using filters and external plugins: https://support.hdfgroup.org/services/filters.html
- ZFP compression example:

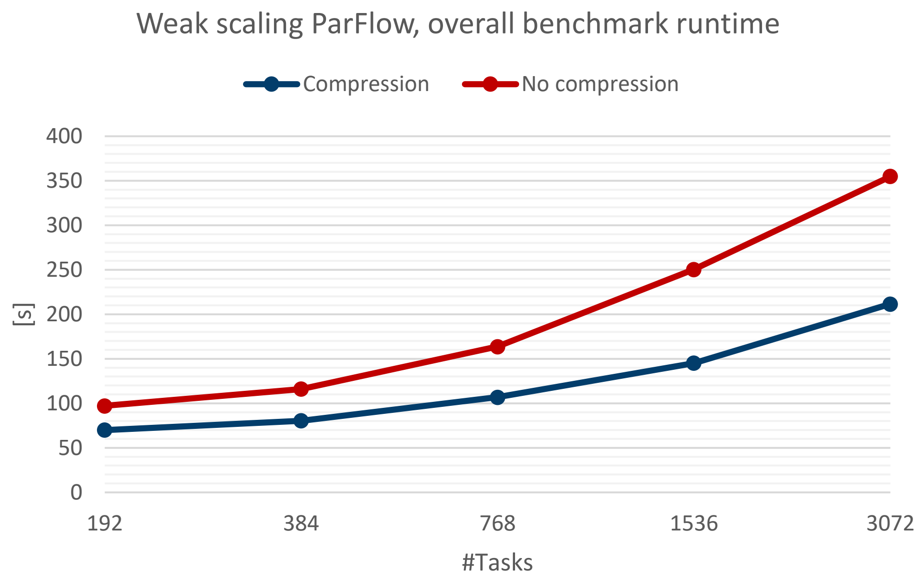
C
```
H5Pset_zfp_reversible_cdata(cd_nelmts, cd_values);
nc_def_var_filter(nc_file_id,nc_variable,H5Z_FILTER_ZFP,
                  cd_nelmts,cd_values);
```

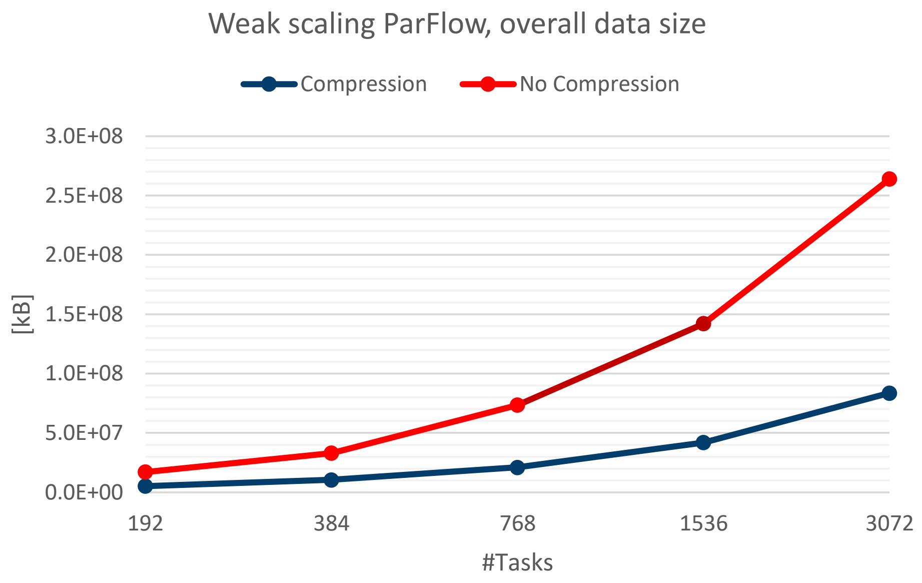
Performance hints

Compression

In-transit data compression:

- HDF5 parallel compression (deflate) capabilities underneath of NetCDF4 were utilized to allow in-transit compression in ParFlow



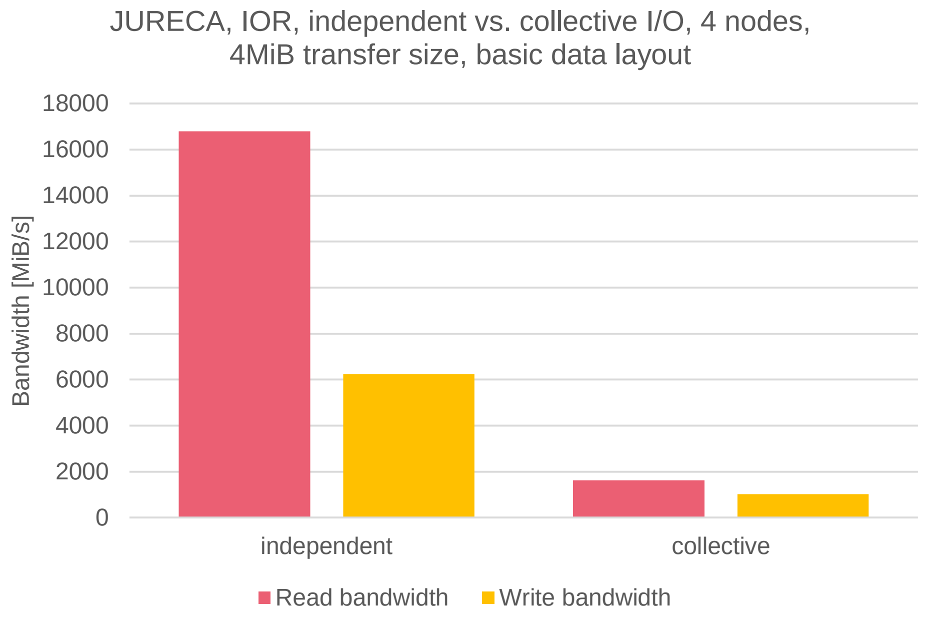


Measurements on JUWELS at JSC

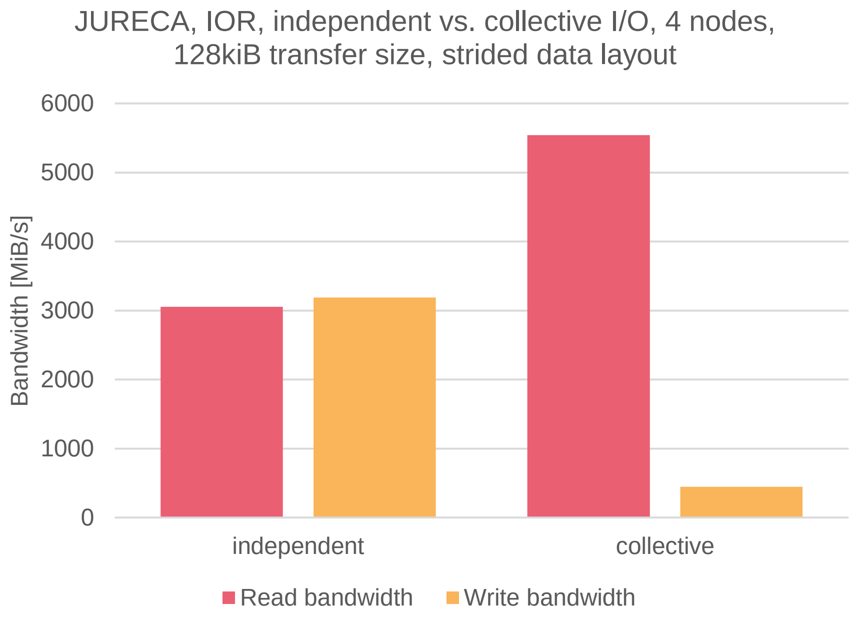
This work was supported by the Energy oriented Centre of Excellence (EoCoE-II), grant agreement number 824158, funded within the Horizon2020 framework of the European Union.

Collective buffering

- Collective I/O operations not always speed up the general I/O, as more data might be processed than needed



	access size [Byte]	count
MPI-IO	4,194,304	184,320
POSIX	16,777,216	264,574



This work was supported by the Energy oriented Centre of Excellence (EoCoE),
grant agreement number 676629,
funded within the Horizon2020 framework of the European Union.

JÜLICH
Forschungszentrum

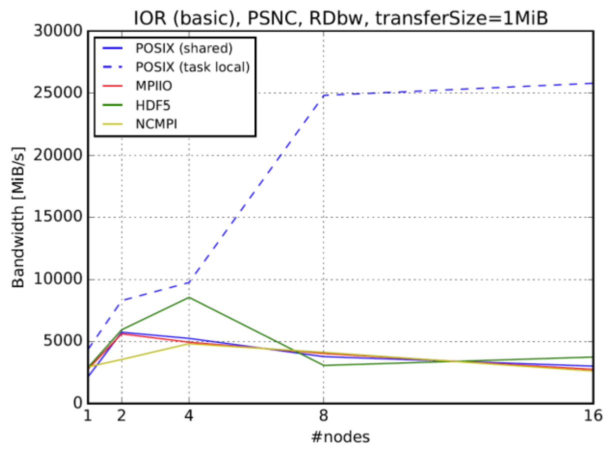
MPI-IO hints

- `romio_cb_read: Enable collective buffering (reading)`
- `romio_cb_write: Enable collective buffering (writing)`
- `cb_buffer_size: Collective buffering, buffer size`
- `cb_nodes: Aggregator nodes`
- `romio_ds_read: Enable data sieving (reading)`
- `romio_ds_write: Enable collective buffering (writing)`

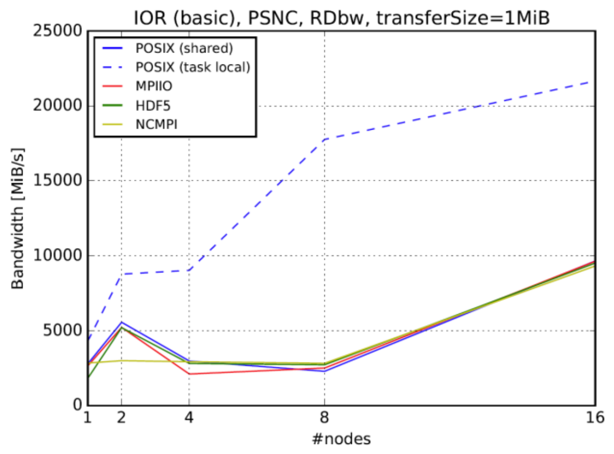
```
export ROMIO_HINTS=romio_hints_file
```

Filesystem specific options

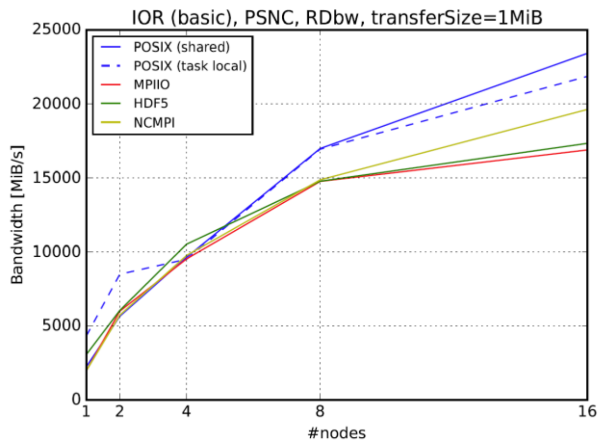
- On Lustre filesystems the user can influence the striping size and the number of involved object storage targets



Default number of OSTs (12) and default strip-size setting (1MiB)



Increased number of OSTs (126)



Increased stripe size to align with the individual amount of data per process (256MiB)

Measurements on Eagle at PSNC

This work was supported by the Energy oriented Centre of Excellence (EoCoE), grant agreement number 676629, funded within the Horizon2020 framework of the European Union.

Profiling with Darshan

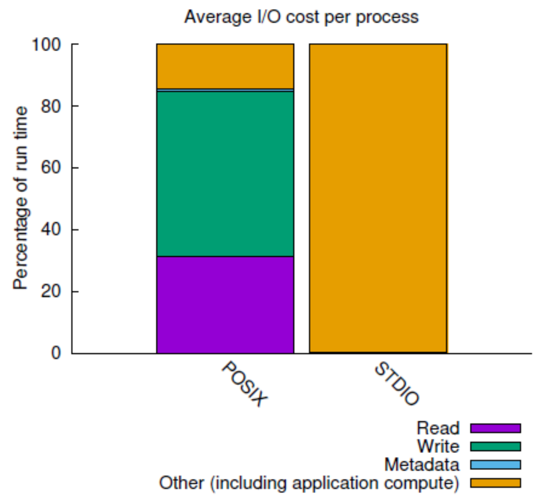
- I/O profiling tool for parallel applications
  - http://www.mcs.anl.gov/research/projects/darshan/
- Integration by using LD_PRELOAD:
  - `LD_PRELOAD=.../lib/libdarshan.so`
- On JUWELS: `DARSHAN_LOG_PATH` points to target log directory
- `DXT_ENABLE_IO_TRACE=1` allows task specific tracing
- Analyse tools:
  - `darshan-parser`: command line access
  - `darshan-dxt-parser`: trace data access
  - `darshan-job-summary.pl`: PDF report
- More details: https://www.mcs.anl.gov/research/projects/darshan/docs/darshan-runtime.html

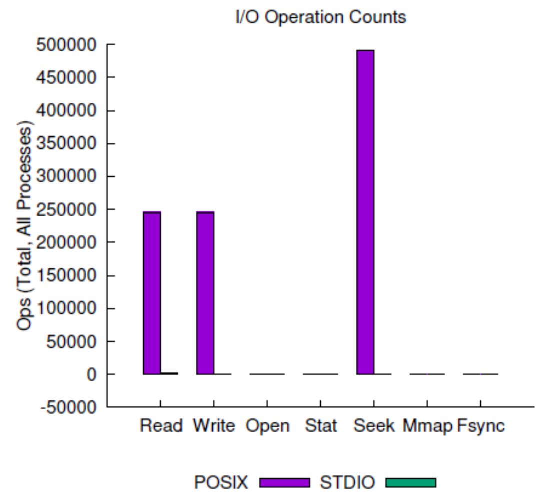
Profiling with Darshan

ior (3/9/2018) 1 of 3

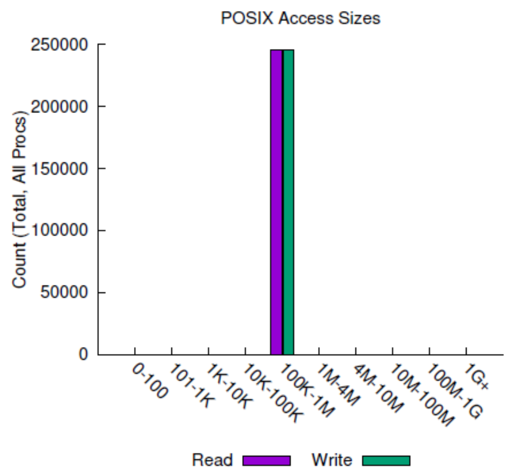
jobid: 4941235	uid: 11901	nprocs: 48	runtime: 10 seconds

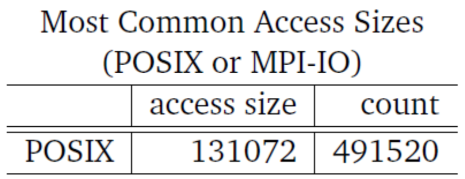
I/O performance *estimate* (at the POSIX layer): transferred 37431 MiB at 6692.22 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.0 MiB at 5.27 MiB/s



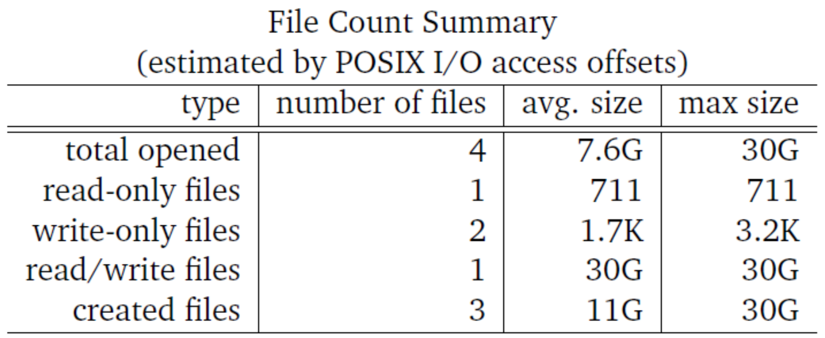


Profiling with Darshan



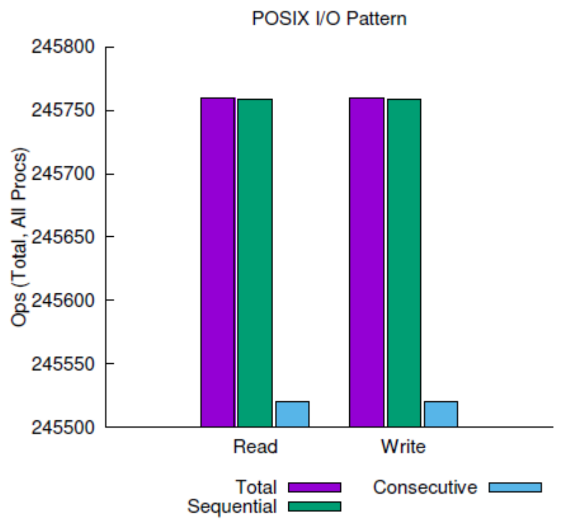

Most Common Access Sizes
(POSIX or MPI-IO)

	access size	count
POSIX	131072	491520


File Count Summary
(estimated by POSIX I/O access offsets)

type	number of files	avg. size	max size
total opened	4	7.6G	30G
read-only files	1	711	711
write-only files	2	1.7K	3.2K
read/write files	1	30G	30G
created files	3	11G	30G

Profiling with Darshan



sequential: An I/O op issued at an offset greater than where the previous I/O op ended.
consecutive: An I/O op issued at the offset immediately following the end of the previous I/O op.

Variance in Shared Files (POSIX and STDIO)

File Suffix	Processes	Fastest			Slowest			σ	
		Rank	Time	Bytes	Rank	Time	Bytes	Time	Bytes
...ehrs/IOR/2_1	48	35	7.507493	1.3G	33	9.180811	1.3G	0.397	0
...or_input.cfg	48	32	0.003404	711	2	0.006366	711	0	0
...<STDOUT>	48	1	0.000000	0	0	0.000392	3.2K	0	455
...<STDERR>	48	1	0.000000	0	0	0.000014	119	0	17

Darshan: Usage example on JUWELS

- Load module
  - `module load darshan-runtime`
- Tell srun to use Darshan (in submit script)
  - `LD_PRELOAD=$EBROOTDARSHANMINRUNTIME/lib/libdarshan.so \ DARSHAN_LOG_PATH=/path/to/your/logdir \ srun … ./executable`
- Analyse output
  - `module load darshan-util`
  - `darshan-job-summary.pl <logfile>.darshan`