

aiXcelerate 2021: Part I – File I/O

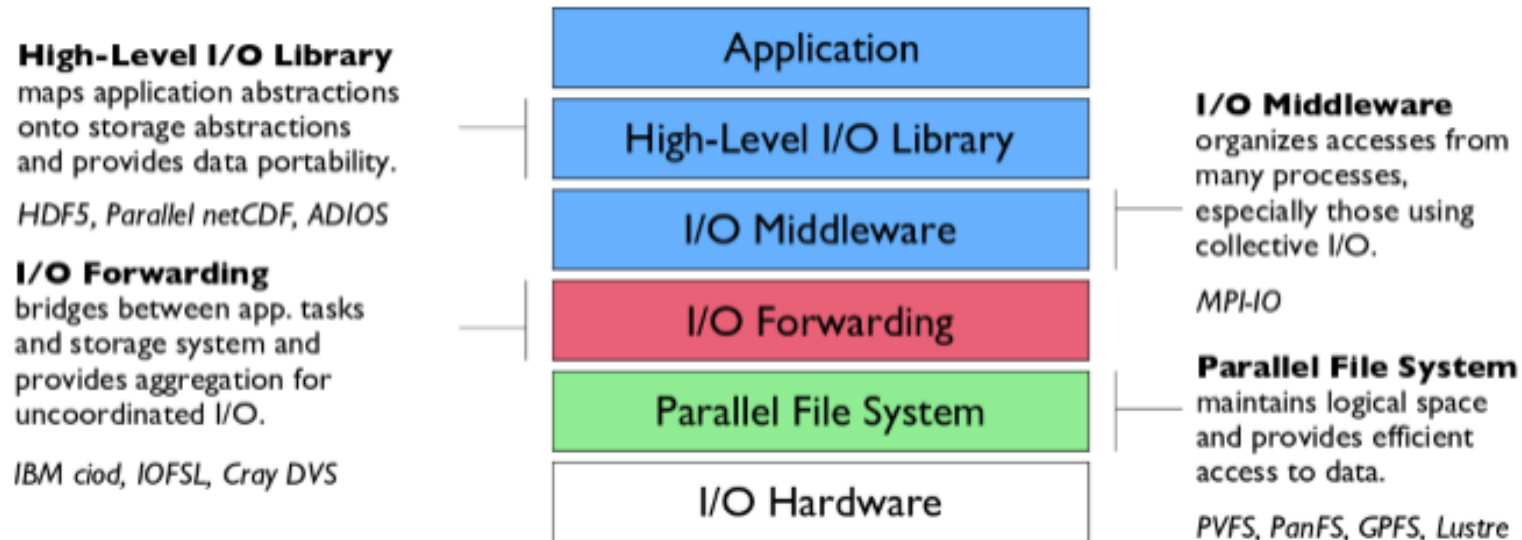
HPC.NRW Competence Network

I/O Analysis with Darshan

Radita Liem (RWTH)

aiXcelerate 2021: Part I – File I/O

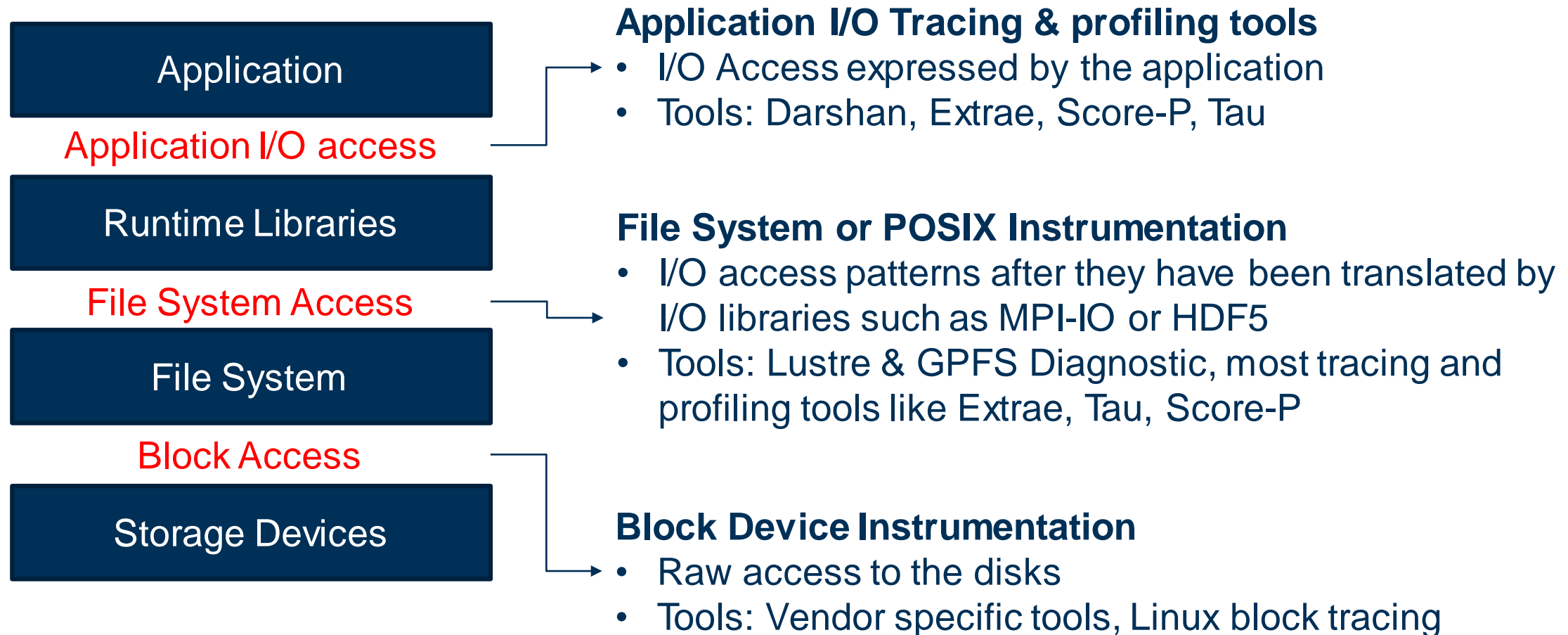
- I/O in HPC Application Overview
- Challenges in I/O measurement
- I/O measurement methods
- Darshan Overview
- Darshan in CLAIX



Additional I/O software provides improved performance and usability over accessing the parallel file system directly. Reduces or (ideally) eliminates need for optimization in application codes.

* This slides are adapted from IISWC 2014, Darshan – I/O Workload Characterization in MPI Applications <https://www.mcs.anl.gov/research/projects/darshan/tutorials/iiswc2014/>

- There are many programming model and patterns that can be used by application to do I/O. It may have many components accessing different files at different time
- I/O performance is affected by the computation and network performance. It's difficult to isolate just I/O
- I/O performance is sensitive to changes in access method, libraries, file systems, and hardware
- I/O performance might be perturbed by the tools used to instrument it



- **Open source runtime library**

- Darshan instrumentation inserted at build time (for static executables) and at run time (for dynamic executables)
- Captures POSIX I/O, MPI-IO, and limited HDF5 and PNetCDF functions

- **Minimal impact on the application**

- Low memory consumption
- Log data collected and compressed at MPI_Finalize() time
- Instrumentation enabled via software modules, environment variables, or compiler scripts
- Doesn't need to change source code or makefile
- No filesystem dependencies

- Compile C, C++, Fortran program that uses MPI.
 - *If it's not an MPI program, put `MPI_Init()` in the beginning of the code and `MPI_finalize()` at the end to enable Darshan generating log file*
 - *The latest version of Darshan doesn't need to use this trick 😊*
- Run the application
- Look for the Darshan log file
 - This will be in a particular directory (depending on your system's configuration)
 - The log file has `.darshan` extension
- Use Darshan commandline tools to analyze the log file
- **Darshan does not capture a trace of all I/O operations:** instead, it reports key statistics, counters, and timing information for each file accessed by the application

- Key tools:
 - **darshan-job-summary.pl** : creates pdf files with graph for initial analysis
 - **darshan-summary-per-file.sh** : similar with above but creates separate pdf file for each file created by Darshan application
 - **darshan-parser**: dumps all information into ascii (text) format
- Python based analysis:
 - Experimental work for Darshan eXtended Tracing : <https://jeanbez.gitlab.io/pdsw-2021/>
- Documentation: <https://www.mcs.anl.gov/research/projects/darshan/docs/darshan-util.html>

– Login to Claix

- `$ module load DEV-TOOLS darshan`

- Below information will show up. This sample is with intelmpi:

```
Use 'export DARSHAN_LOGPATH=<path>' to define experiment directory
Set LD_PRELOAD=/rwthfs/rz/SW/UTIL/perf-
tools/darshan/3.1.8/intelmpi2018.4.274/lib/libdarshan.so to enable measurement
```

– Use above information to generate Darshan measurement in your job script

- `export DARSHAN_LOGPATH=<path>`

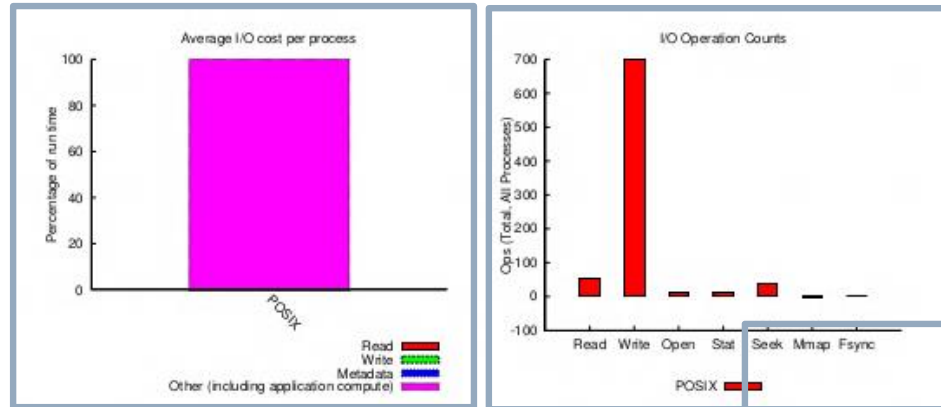
- `export LD_PRELOAD=/rwthfs/rz/SW/UTIL/perf-
tools/darshan/3.1.8/intelmpi2018.4.274/lib/libdarshan.so`

- Run the application, e.g: `$MPIEXEC some.binary`

- In your designated `DARSHAN_LOGPATH` directory you will find new file with `.darshan` extension. Run:
 - `$ darshan-job-summary.pl *.darshan`
- In your current directory you will find new pdf file with the same name as the darshan log file
 - If the generation stucks, exit from the program (ctrl + c). Most of the time, it is because of several LaTeX components are not available in your environment and you need to load texlive module
 - `$ module load MISC texlive`
 - Repeat the process
- Open the pdf file

Darshan in CLAI - Interpreting Darshan Report (1)

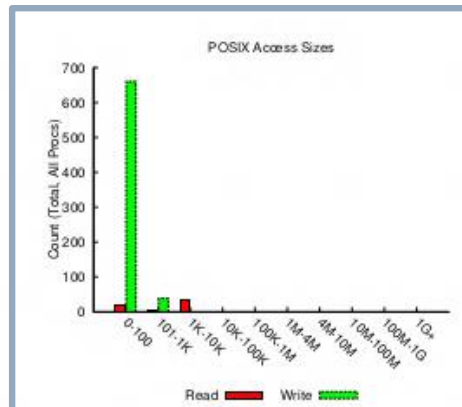
I/O performance estimate (at the POSIX layer): transferred 0.2 MiB at 9.74 MiB/s



Shows the amount of data transferred and what the bandwidth was. The bandwidth data could be used to determine the I/O scalability of the application at different MPI process counts to determine if I/O is scaling linearly

Percentage of I/O in the runtime

I/O operation counts. Ideally, the operation counts should be small as possible as a large number could indicate a performance issue



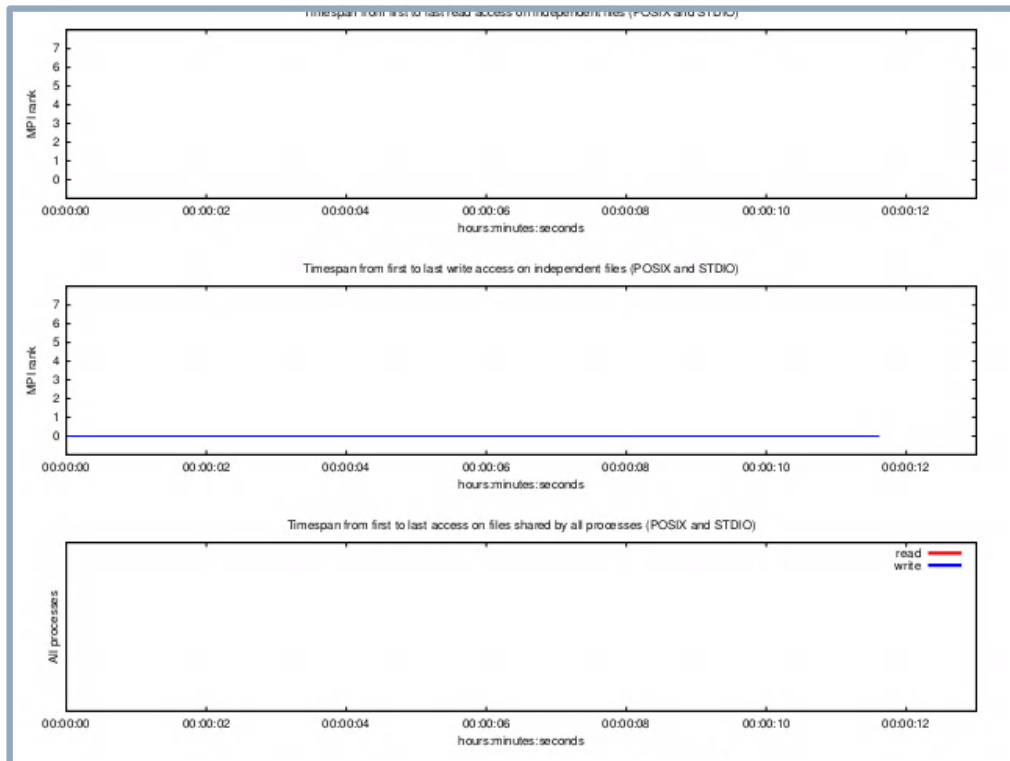
Read/Write access size chart. Ideally, the number of I/O operations should be low with large access sizes

Darshan in CLAIx - Interpreting Darshan Report (2)

Most Common Access Sizes (POSIX or MPI-IO)			File Count Summary (estimated by POSIX I/O access offsets)			
	access size	count	type	number of files	avg. size	max size
POSIX	48	174	total opened	3	13K	27K
	32	88	read-only files	1	551	551
	37	88	write-only files	1	27K	27K
	51	87	read/write files	1	9.3K	9.3K
			created files	2	19K	27K

The tables give a breakdown of the access sizes, and the table on the right shows the file count summary.

Ideally, the file operation counts should be low, particularly the write-only files.



I/O timeline. Can see only write or only read timeline as well as aggregated timeline. In this case, nothing to see because the size is too small

Darshan in CLAIx - Interpreting Darshan Report (3)

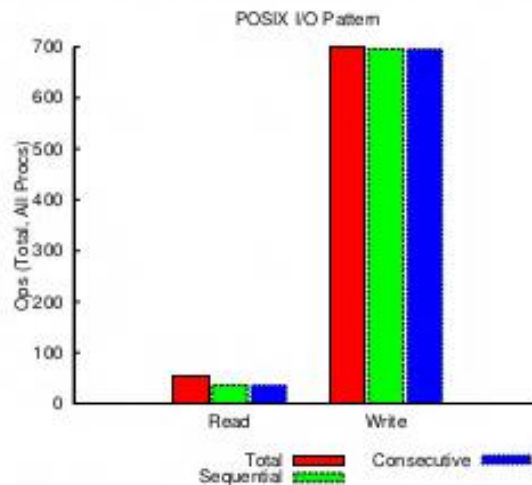
Average I/O per process (POSIX and STDIO)

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	1.025e-05	0.000131368637084961
Independent writes	0.00204075	0.00327908992767334
Independent metadata	0.00011225	N/A
Shared reads	0.000100125	0.0181560516357422
Shared writes	2.9625e-05	0.00113475322723389
Shared metadata	0.000577875	N/A

The table shows detailed I/O process and the filesystem data transfer speed. Both are useful to determine whether the application is running the correct file system

Data Transfer Per Filesystem (POSIX and STDIO)

File System	Write		Read	
	MiB	Ratio	MiB	Ratio
/rwthfs/rz/cluster/hpcwork/qi190786	0.03531	1.00000	0.14630	1.00000



I/O pattern breaks down the consecutive & contiguous I/O components

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
...f/tea.in.tmp	8	1	0.000548	19K	0	0.001340	28K	0	3.15e+03

The last table shows the variance of POSIX I/O data across the MPI processes. Can be used to determine if there are any load imbalances across the MPI processes

- The easiest way to profile an application with Darshan is to build a dynamic executable (executable that is dynamically linked with the MPI library). To determine if your executable is dynamic or not, type:

```
$ ldd <executable>
```
- If the output shows the MPI library, then it is dynamically linked.
- Set the Darshan log path variable and put LD_PRELOAD in front of the MPI execution command with the Darshan library :

```
$ export DARSHAN_LOGPATH=.  
$ LD_PRELOAD=<follow notes in loaded module> <mpiexec command>  
<executable>
```
- A trace file will be created in the current working directory. The trace file name should start with the user name and has .darshan extension

- For static executable, new compile script needs to be generated to include darshan.
- Some examples for MPICH based MPI library:
 - \$ darshan-gen-cc.pl `which mpicc` --output mpicc.darshan
 - \$ darshan-gen-cxx.pl `which mpicxx` --output mpicxx.darshan
 - \$ darshan-gen-fortran.pl `which mpif77` --output mpif77.darshan
 - \$ darshan-gen-fortran.pl `which mpif90` --output mpif90.darshan
- For other types of libraries, can refer to the darshan-get-* files or contact the Darshan support mailing list

- `DARSHAN_DISABLE`: disables Darshan instrumentation
- `DARSHAN_INTERNAL_TIMING`: enables internal instrumentation that will print the time required to startup and shutdown Darshan to stderr at run time.
- `DARSHAN_LOGHINTS`: specifies the MPI-IO hints to use when storing the Darshan output file. The format is a semicolon-delimited list of `key=value pairs`, for example: `hint1=value1;hint2=value2`
- `DARSHAN_MEMALIGN`: specifies a value for system memory alignment
- `DARSHAN_DISABLE_SHARED_REDUCTION`: disables the step in Darshan aggregation in which files that were accessed by all ranks are collapsed into a single cumulative file record at rank 0. This option retains more per-process information at the expense of creating larger log files. Note that it is up to individual instrumentation module implementations whether this environment variable is actually honored.
- `DARSHAN_LOGPATH`: specifies the path to write Darshan log files to. Note that this directory needs to be formatted using the `darshan-mk-log-dirs` script.
- `DARSHAN_ENABLE_NONMPI`: setting this environment variable is required to generate Darshan logs for non-MPI applications

- **DARSHAN_LOGFILE**: specifies the path (directory + Darshan log file name) to write the output Darshan log to. This overrides the default Darshan behavior of automatically generating a log file name and adding it to a log file directory formatted using darshan-mk-log-dirs script.
- **DARSHAN_MODMEM**: specifies the maximum amount of memory (in MiB) Darshan instrumentation modules can collectively consume at runtime (if not specified, Darshan uses a default quota of 2 MiB).
- **DARSHAN_EXCLUDE_DIRS**: specifies a list of comma-separated paths that Darshan will not instrument at runtime (in addition to Darshan's default blacklist)
- **DXT_ENABLE_IO_TRACE**: setting this environment variable enables the DXT (Darshan eXtended Tracing) modules at runtime for all files instrumented by Darshan. Currently, DXT is hard-coded to use a maximum of 4 MiB of trace memory per process (in addition to memory used by other modules).
- **DXT_DISABLE_IO_TRACE**: setting this environment variable disables the DXT module at runtime for all files instrumented by Darshan.

- Beware of the compiler combination when using Darshan, some compiler combinations are not supported in the CLAIX modules
- HDF5 Support
 - Current Darshan module doesn't support HDF5 profiling, create a new installation `--enable-hdf5-mod=/path/to/hdf5/`
 - The default report generator tools doesn't show the HDF5 output
- Machine learning application usually produces a lot of files. Consider filtering the files using: `DARSHAN_EXCLUDE_DIRS` variable
 - There's a limit of 1024 files per process in Darshan that can't be changed. If the application hits the limit, the rest of the files will not be recorded.