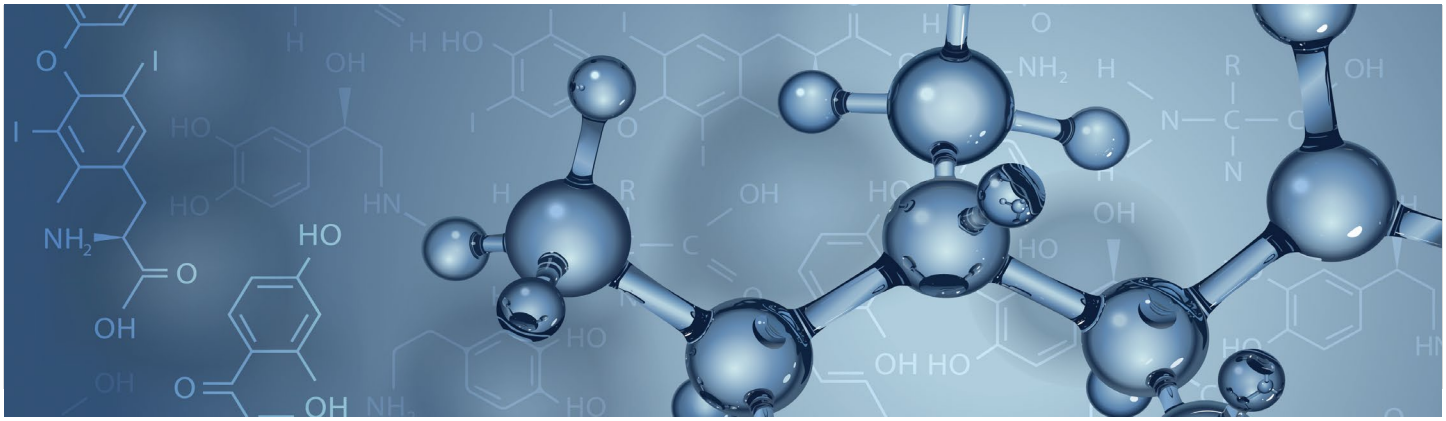
I/O PROFILING TO IMPROVE DL_POLY FOR MOLECULAR DYNAMICS SIMULATION

Liam McClean, Senior Software Engineer, Altair / Aidan Chalk, HPC Engineer, STFC Hartree Centre / March 26, 2021



Supporting Research and Industry in the UK

We worked with the team at the Science and Technology Facilities Council (STFC) Hartree Centre to assess and improve the performance of a number of commonly used high-performance computing (HPC) applications. The Hartree Centre provides some of the most advanced HPC, data, and AI technologies in the world to support UK research and industry.

Recently, the team at the Hartree Centre used Altair Breeze™ to profile and improve DL_POLY, a general-purpose classical molecular dynamics (MD) simulation software developed at STFC's Daresbury Laboratory. This paper presents the initial findings and performance improvements that have been submitted to the DL_POLY development repository.

By looking at the I/O patterns using Breeze the Hartree Centre was able to reduce simulation software run time by at least 8% with a relatively small investment of time.

About DL_POLY

DL_POLY provides scalable performance from a single processor workstation to a high-performance parallel computer, allowing the dynamic simulation of very large systems of atoms and molecules. It can be compiled as a serial application code, using only a Fortran 90 compiler, or as a parallel application code, provided an MPI2 instrumentation is available on the parallel machine.

About Breeze

Detailed dependency analysis and I/O profiling with Breeze makes every engineer an I/O expert. Breeze users can quickly solve software deployment problems and resolve file and network dependencies. With detailed data for storage exports and summary reports for sharing, Breeze identifies good and bad I/O for easy wins and profiles application file I/O to ensure files are stored in the right place.

The Initial Trace

The team at the Hartree Centre performed an initial trace with Breeze for a slightly modified version of the Sodium Chloride test case provided with DL_POLY_4. They doubled the size of the example in each dimension using the nfold 2 2 2 option, then forced DL_POLY to use the sorted mpiio writing scheme, with 1 writer per node (16 total writing ranks).

During the profile, the team also simulated a high-I/O run used by some of the DL_POLY user base by adding traj 0 1 2 to the CONTROL file, and they ran the simulation for 1,250 steps.

Since the team ran the application for 1,250 steps, they expected each node to only open the HISTORY file 1,251 times, once for the initial setup then once per timestep. However, the MPI tab in Breeze showed that 40,032 calls were made to MPI_Open() for the HISTORY file on each node. Each node is shown as a separate child trace. This took more than 17min 45s on the head node, which is

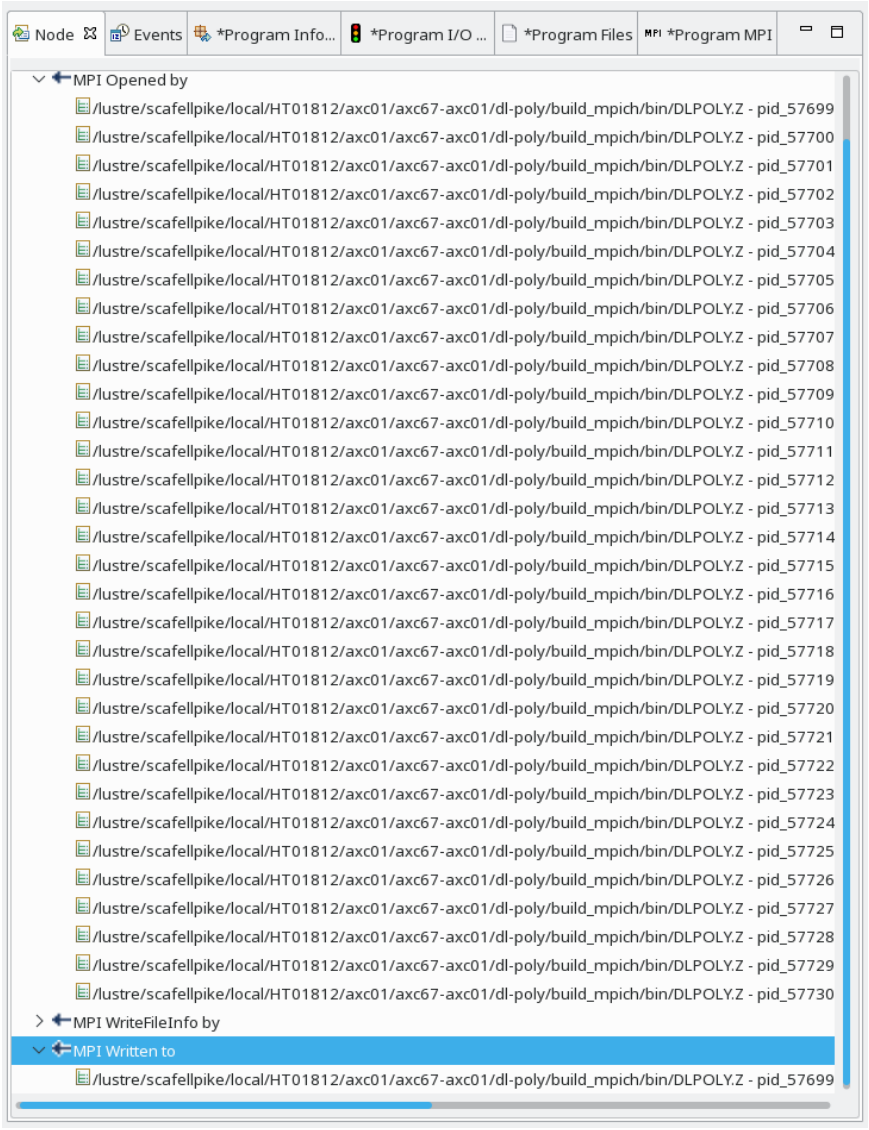
a lot of wait time on a run that took only 14 mins. The time is split between 32 ranks, so each rank had on average 33s of wait time on those opens. Other nodes saw similar wait times.

Mount Point	Location	MPI Open				MPI ...	MPI Stat	MPI Sync
Mount Point	Filename	# Call	Total latency (µs)	Max latency (µs)	# Failure	# Call	# Call	# Call
/	REVCON	416	10s 595ms 174µs	31ms 90µs	0	416	0	0
/	HISTORY	40,032	17min 45s 97ms 96...	133ms 523µs	0	40,032	0	0
/	CONFIG	32	35s 414ms 43µs	1s 106ms 744µs	0	32	0	0

Breeze Screenshot: The HISTORY File Was Opened More Than 40K Times With an I/O Wait Time of Over 17 Minutes

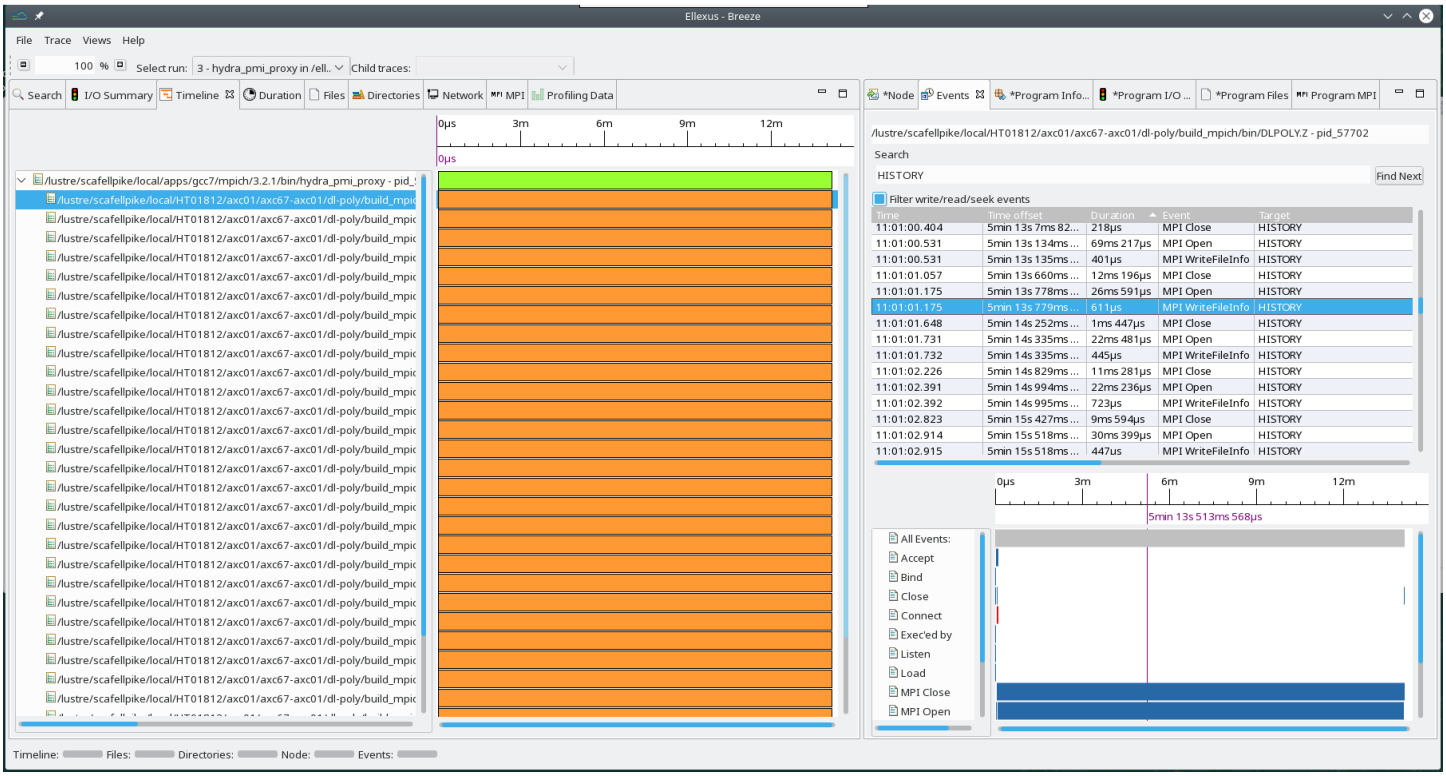
Making Improvements

DL_POLY already uses a separate MPI communicator to perform MPI_write() operations but does not use this communicator for opening the files. This is clearly seen in the screenshot below where Breeze shows that all ranks open the HISTORY file, but only one writes to it.



Breeze Screenshot: The HISTORY File Was Opened by Every MPI Rank, but Only One Rank Wrote to It

Checking each rank to verify this behavior shows that most ranks simply open the file, modify the metadata, and close it once for each time step.



Breeze Screenshot: Most MPI Ranks Opened the HISTORY File, Modified the Metadata, Then Closed It

The Hartree Centre team modified the io, trajectory, and configuration modules in DL_POLY to enable only ranks that perform MPI_write() operations to perform the MPI_Open() calls.

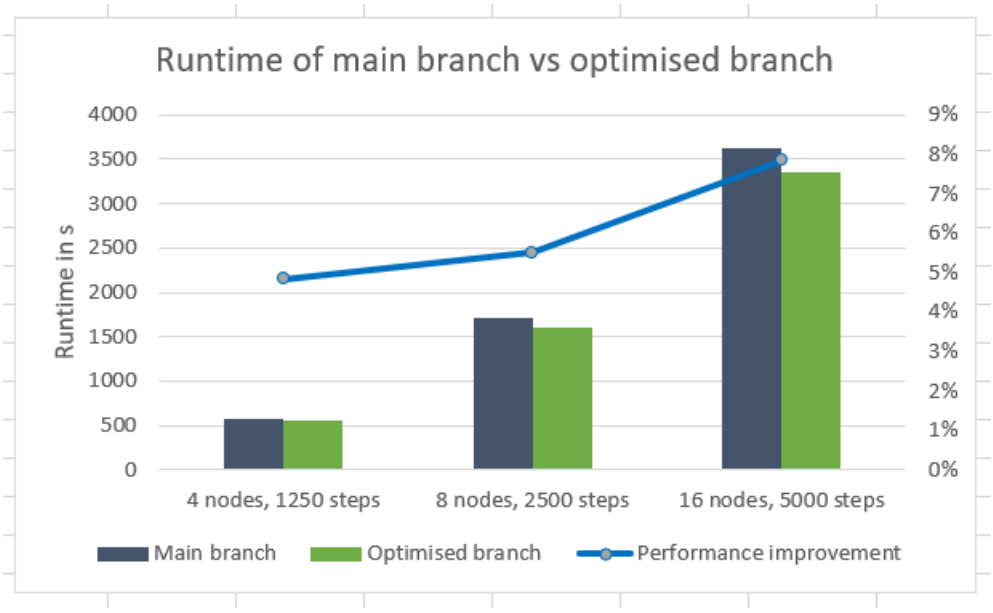
Retracing this in Breeze showed the expected improved results:

Location	MPI Open	MPI Filesystem change
Filename	# Call	# Call
HISTORY	1251	1251
CONFIG	32	32
REVCON	13	13

Breeze Screenshot: Improved I/O Patterns Showing the Expected Number of Open Calls

Even Better Results in Scaling

Furthermore, when run for 5,000 steps instead of 1,250 on 512 ranks (16 nodes), the run time of this example went from 3632.781s to 3347.844s, averaged over three runs. This is a performance improvement of around 8%. The example showed similar performance gains of 5-6% when run on 4 or 8 nodes respectively. The performance improvements are likely to be greater when running at larger scales.

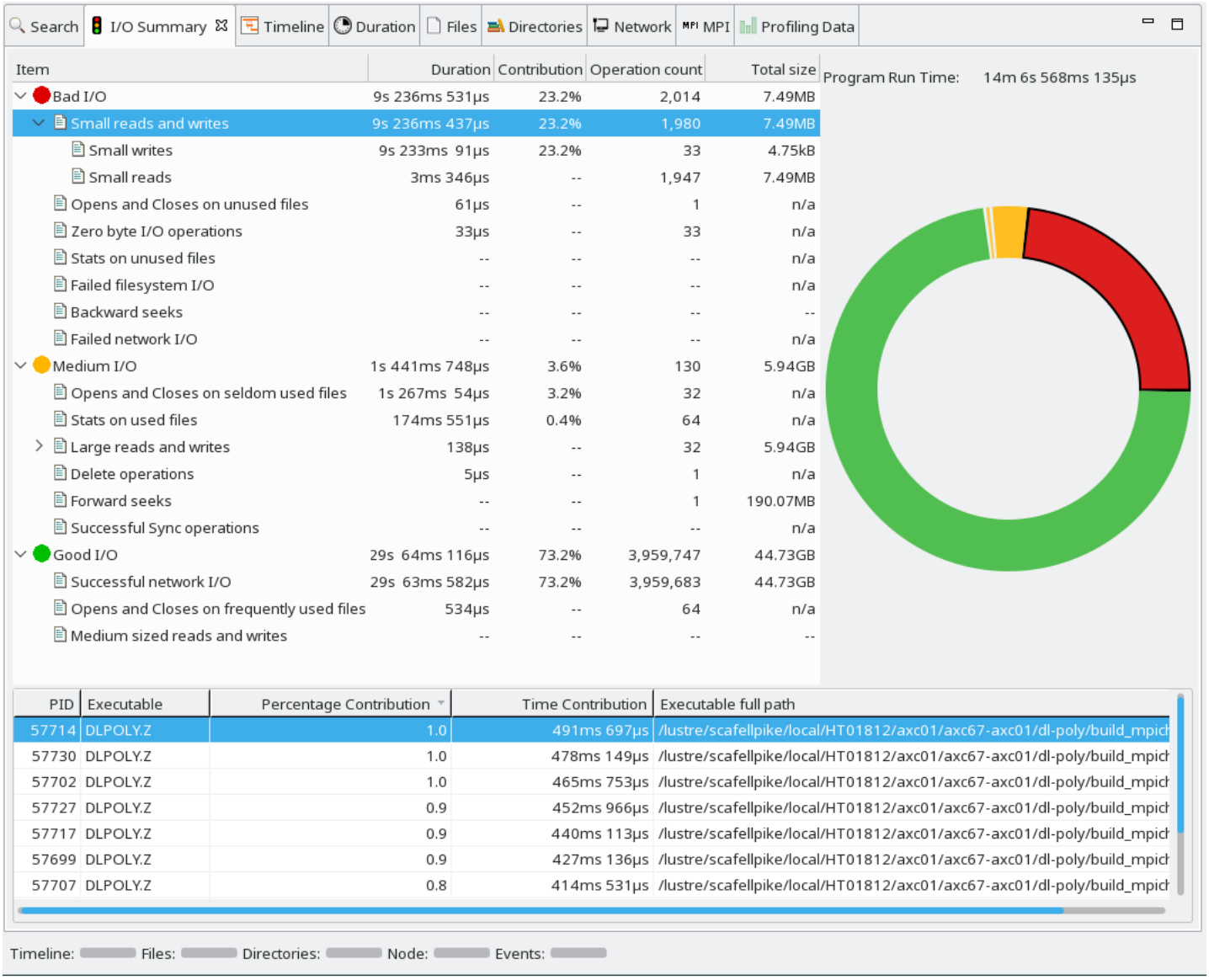


Performance Improvements Are More Pronounced as the Application Scales

These changes have been submitted to the DL_POLY development repository, and the team expects to see the improvements as part of the next major release.

Further Improvement

Although unlikely to lead to such impressive gains, there are potentially some small wins to be had in optimizing the writes to the /netfs file system. Within the 14min run time there was just over 9s of small writes to an OUTPUT file on that mount point. With only 32 writes performed and few bytes written each time, there is clearly scope for moving that I/O into a separate thread to shave off another 9s from the application run time.



Item	Duration	Contribution	Operation count	Total size
Bad I/O	9s 236ms 531µs	23.2%	2,014	7.49MB
Small reads and writes	9s 236ms 437µs	23.2%	1,980	7.49MB
Small writes	9s 233ms 91µs	23.2%	33	4.75kB
Small reads	3ms 346µs	--	1,947	7.49MB
Opens and Closes on unused files	61µs	--	1	n/a
Zero byte I/O operations	33µs	--	33	n/a
Stats on unused files	--	--	--	n/a
Failed filesystem I/O	--	--	--	n/a
Backward seeks	--	--	--	--
Failed network I/O	--	--	--	n/a
Medium I/O	1s 441ms 748µs	3.6%	130	5.94GB
Opens and Closes on seldom used files	1s 267ms 54µs	3.2%	32	n/a
Stats on used files	174ms 551µs	0.4%	64	n/a
Large reads and writes	138µs	--	32	5.94GB
Delete operations	5µs	--	1	n/a
Forward seeks	--	--	1	190.07MB
Successful Sync operations	--	--	--	n/a
Good I/O	29s 64ms 116µs	73.2%	3,959,747	44.73GB
Successful network I/O	29s 63ms 582µs	73.2%	3,959,683	44.73GB
Opens and Closes on frequently used files	534µs	--	64	n/a
Medium sized reads and writes	--	--	--	--

PID	Executable	Percentage Contribution	Time Contribution	Executable full path
57714	DLPOLY.Z	1.0	491ms 697µs	/lustre/scafellpike/local/HT01812/axc01/axc67-axc01/dl-poly/build_mpich
57730	DLPOLY.Z	1.0	478ms 149µs	/lustre/scafellpike/local/HT01812/axc01/axc67-axc01/dl-poly/build_mpich
57702	DLPOLY.Z	1.0	465ms 753µs	/lustre/scafellpike/local/HT01812/axc01/axc67-axc01/dl-poly/build_mpich
57727	DLPOLY.Z	0.9	452ms 966µs	/lustre/scafellpike/local/HT01812/axc01/axc67-axc01/dl-poly/build_mpich
57717	DLPOLY.Z	0.9	440ms 113µs	/lustre/scafellpike/local/HT01812/axc01/axc67-axc01/dl-poly/build_mpich
57699	DLPOLY.Z	0.9	427ms 136µs	/lustre/scafellpike/local/HT01812/axc01/axc67-axc01/dl-poly/build_mpich
57707	DLPOLY.Z	0.8	414ms 531µs	/lustre/scafellpike/local/HT01812/axc01/axc67-axc01/dl-poly/build_mpich

Breeze Screenshot: Small Read and Write Operations Highlighted in Breeze I/O Summary

Conclusion

This collaborative work with the HPC experts at the Hartree Centre demonstrates how I/O profiling can reveal easy wins with minimal effort. Improving run time and overall performance doesn't have to involve a complete rewrite of an application; often, only a small change is needed. Knowing where to look is the clever part.

Acknowledgements

This work was funded by a Bridging for Innovators (B4I) grant. B4I offers businesses unique access to a suite of high-tech scientific facilities and knowledge to fast-track solutions to industrial challenges.