The Need for Parallel I/O in Classical Molecular Dynamics

I.T. Todorov, I.J. Bush & A.R. Porter
Advanced Research Computing, CSED
STFC Daresbury Laboratory, UK

ABSTRACT: *We discuss the need for parallel I/O in classical molecular dynamics (MD) simulations. While reading is a one off operation during MD simulations, writing is a fairly repetitive one which can be prohibitively pricy time-wise with respect to the actual compute time per MD timestep. Comparisons between different writing strategies on a variety of modern platform are made and discussed. Despite that ASCII format is spacious and compromises on precision it is still the fastest alternative there is. While MPI-I/O can be a fast alternative it is not widely available and well supported. NetCDF format is a promising alternative to ASCII, it is not yet utilising the MPI-I/O functionality and its serial I/O is prohibitively slow.*

KEYWORDS: DL_POLY_3, Molecular Dynamics, Parallel I/O, Fortran Direct Access, MPI I/O, NetCDF

1. Introduction

DL_POLY_3 [1] is a general purpose molecular dynamics (MD) package developed by I.T. Todorov and W. Smith at STFC Daresbury Laboratory to support researchers in the UK academic community [2]. This software is designed to address the demand for large-scale MD simulations on multiprocessor platforms, although it is also available in serial mode. DL_POLY_3 is fully self-contained and written in FORTRAN 95 in a modularised manner with communications handled by MPI2. The standards conformance of the code has been very rigorously checked using the NAGWare95 and FORCHECK95 analysis tools, so guaranteeing exceptional portability. Parallelisation is achieved by equi-spatial domain decomposition distribution which guarantees excellent load balancing and full memory distribution provided the particle density of the system is fairly uniform across space [3]. This parallelisation strategy results in mostly point to point communication with very few global operations [4], and excellent scaling [5]; one might compare it with the halo exchange algorithms employed in computational fluid dynamics.

Although parallelisation of the computation is very important, for DL_POLY_3 to be an effective tool for researchers all parts of the calculation must scale. In particular, this includes the input and output (I/O) stages of the code. Historically, this has most often been performed in an essentially serial manner, not only in DL_POLY_3 but also in many other large scale packages. However, with the scale of calculations now possible, this approach will not scale to the next generation of machines as the time taken by I/O is becoming prohibitive.

An example of the problem is a relatively recent calculation performed on an oxygen deficient pyrochlore on the BG/L at Jülich, Germany [6]. The compute scaled well on this large system (around 15,000,000 particles), with the wall-clock time per time step decreasing from 2.7 seconds on 2048 processors to 0.49 on 16384. However, it was not possible to perform any science on this system simply because the (serial) I/O was too expensive, taking over 10 minutes to dump the atomic positions (the atomic *configuration)*. As this needed to be done roughly every 1,000 time steps, it is clear that the good scaling of the parallel compute portion of the code is overshadowed by the serial I/O.

Historically, all I/O in DL_POLY_3 has been in ASCII. A simple and fast solution to the I/O bottleneck would be to move to Fortran binary I/O. Unfortunately, this is not practical for the DL_POLY user base as usually simulations are performed on a supercomputer whereas the analysis of the results is performed on a workstation at the user's home institution. Thus the output files must be portable making ASCII a good choice were it not for the performance issues noted above. This need for both portability and performance has caused us so far to make preliminary investigations into two approaches:

1. Parallel ASCII I/O
2. Serial NetCDF

and this paper presents our preliminary results. As the majority of the I/O time is in writing rather than reading we shall only consider the former.

2. Parallel ASCII I/O

In DL_POLY_3.09, two conceptual solutions to ordered parallel ASCII printing exist in three principal implementations.

1. Serial direct access write (abbreviated as SDAW) – where only a single node, the master, prints it all and all the rest communicate information to a master in turn while the master completes writing a configuration of the time evolution.
2. Parallel direct access write (PDAW) – where all nodes print in the same file in an orderly manner so no overlapping occurs using Fortran direct access files. However, it should be noted that the behaviour of this method is not defined by the Fortran standard, and in particular we have experienced problems when disk cache is not coherent with the memory.
3. Finally, MPI-I/O write (MPIW) which has the same concept as the PDAW but is performed using MPI-I/O rather than direct access.

To test the I/O the system we used was an oxygen deficient pyrochlore $Gd_2Zr_2O_7$ (zirconite) consisting of 3,773,000 particles, which requires a 1.1 GB configuration dump file. It is worth mentioning that no machine was available for exclusive use while benchmarking, which could have contributed to the fluctuations of the observed times at low processor counts. The parallel timings presented are from runs undertaken on the following platforms:

- The UK National Supercomputing facility HPCx [7], sited at STFC Daresbury [8], comprising of 160 IBM p5-575 nodes, totalling 2560 POWER5 1.5 GHz compute cores.
- The BG/L, 1024 PowerPC 440 700 MHz compute cores, and BG/P, 4096 PowerPC 450 850 MHz compute cores, clusters [9] sited at STFC Daresbury [8].
- The UK National Supercomputing facility HECToR [10], sited at the University of Edinburgh, with 60 Cray XT4 cabinets totalling 11,328 AMD 2.8 GHz Opteron compute cores.
- The Swiss Supercomputing Centre CSCS [11], comprising of 18 Cray XT3 cabinets, totalling 3328 AMD 2.6 GHz Opteron compute cores.

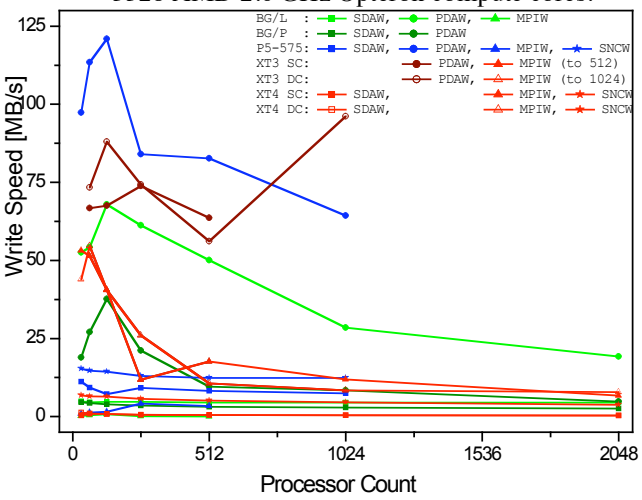

Figure 1

Plotted in Figure 1 is the write speed in MBytes per second with increasing processor count. The notation specifies the type of platform followed by the type of writing method. The benchmarks on the Cray XT3/4 platforms were run in two modes: single core (SC) mode – when only a single core per dual core socket is engaged, and dual core (DC) mode – when both cores of the socket are engaged. Except where noted, all of the runs were done with the default settings for any environment variables.

It can be seen that, where results are presented, the performance ordering of the different methods is the same on all of the machines examined: PDAW (represented by circles) is the most efficient, MPIW (triangles) the next and the SDAW (squares) the least. Cases where the runs were not successful are reflected by gaps in the data with reasons as outlined below. However, it is very clear that while the scaling is poor, a marked increase in performance can be achieved through parallel I/O.

The IBM and Cray platforms use different file systems; BG/L, BG/P and P5-575 employ GPFS, while the two Cray systems, the XT3 and XT4, have Lustre. While all GPFS systems guaranteed cache coherency between

memory and disk, this was not so for Lustre. Interestingly, the Cray XT3 running Catamount with Lustre had no problem with cache coherency, i.e. see PDAW data, but when SDAW was employed it performed so poorly[1] that no performance data is presented. However, the opposite was found on the Cray XT4 running CNL where SDAW performed correctly but PDAW was not successful as spurious NULL characters were introduced into the output stream. For BG/P the MPIW benchmarks did not complete within 5 hours and so again no performance data is presented. Further, MPIW also failed on the highest processor counts of BG/L and P5-575.

It can be seen that the PDAW algorithm was performing markedly superiorly to the SDAW or MPIW methods when supported by the platform for this particular size of messages (73 Bytes ASCII per message). Improvements by an order of magnitude can be obtained, and it is clear that if regular writing is required when studying a system that this parallel strategy will markedly improve the scaling of the whole code by reducing the time for I/O relative to that required for the compute, *even though the I/O is not scaling especially well itself*.

It is also worth noting that for all the benchmarked platforms, it was only the Cray XT3/4 where the MPI-I/O write performed consistently well and much better than the standard serial direct access write. However, as seen on the Cray XT3, this was still not as fast as the parallel direct access method. Interestingly, when the number of object storage targets (OSTs) being used on the Lustre system was optimised to give the best performance of MPIW for the simulated system, an improvement by a factor of two was observed. Such an improvement means that MPIW can achieve similar performance to PDAW on the Cray XT3 once the storage methodology is optimised for the dedicated I/O processing units.

Comparing the fastest write strategies on all platforms, we see that those from IBM (BG/P/L and P5-575) deliver the best I/O performance when running on 128 compute cores, whereas the Cray XT3/4 both perform best on 32 compute cores in single core mode or 64 in dual core mode.

[1] On the XT3 the SDAW method took over 9 hours to complete. This might have been a consequence of the use of the CRAY XT3 Catamount microkernel, which is known to implement ASCII printing in a character by character manner.

3. Serial NetCDF I/O

It has been long known that I/O in native binary data is faster and more disk-compact than ASCII and data precision is not compromised as is the case with data in ASCII. Despite these advantages, binary data is awkward to work with because it is not portable across different platforms. Unfortunately, as noted above, the analysis of DL_POLY_3 runs is often performed on a different machine to that on which the code originally executed, so plain binary is not acceptable. However, this significant drawback has been addressed by the XDR standard in a number of portable binary implementations, one of which, NetCDF (network Common Data Form) [12], has become dominant. NetCDF provides a set of software libraries and machine-independent data formats for array-based, scientific data and is widely used by various scientific communities. The current stable release of this software does not support parallel I/O but the next release (currently in beta) will, through the use of MPI-I/O. We have very recently implemented serial, NetCDF-based I/O[2] within DL_POLY_3 and shall refer to this as the SNCW (Serial NetCDF Write) algorithm, and here present our preliminary findings.

When SDAW was forced to native binary rather than ASCII a speed-up by a factor of 2.5 to 3 was seen in all benchmarks on all machines. This relates very well to the ratio of the size of the messages sent per record in either case – 73 ASCII characters (73 Bytes) to three 64 bit floating point numbers (3×8 Bytes = 24 Bytes). However, when SNCW was tested the speed-up varied significantly between the two platforms that were benchmarked. While on the IBM P5-575 SNCW was only 1.5 times faster than SDAW on average, on the Cray XT4 it was 10 times and it did not matter whether runs were in single- or dual-core mode. *Despite these differences, SNCW performed 2.5 times faster on the IBM P5-575 than on the Cray XT4*.

4. Conclusion

We have demonstrated that the use of a parallel I/O strategy in writing MD configuration files during simulations can bring a dramatic improvement in the overall performance by markedly reducing the time for the I/O compared to the compute, *even though the I/O is not scaling especially well itself*. It is clear from the discussion above that for DL_POLY_3 to be able to

[2] The resulting trajectory files can conform to the NetCDF formats used by the Molecular Modelling Tool Kit [13] or Amber [14]

address larger systems we must use a parallel I/O strategy, for otherwise the I/O will dominate any compute.

For the NetCDF strategy, while SNCW provides performance and storage benefits over the SDAW algorithm, it will ultimately be limited due to its lack of parallelism. We, therefore, intend to look at other possible solutions, such as a parallel version of the SNCW strategy once the appropriate software is available. We hope that the combination of our two strategies will bring the benefits of both, and indeed it should be noted that the order of magnitude improvement in performance in I/O is the bare minimum required to make scientific studies on large systems practical, and that at least a further factor of 2 would be desirable. Further, we must also note that parallel NetCDF will depend on MPI-I/O, and for our desired performance to be achieved on all platforms the problems with MPI-I/O that we have found must be addressed.

Acknowledgments

The authors would like to thank David Quigley at the University of Warwick and Martyn Foster and Lucian Anton at NAG UK for their help and useful discussions.

About the Authors

Ilian Todorov, Ian Bush (*e*mail: *I.J.Bush@dl.ac.uk*) and Andrew Porter are HPC engineers at STFC Daresbury Laboratory in Cheshire, UK. Ilian Todorov is an application developer and co-author of the DL_POLY package. Ian Bush is the author of Daresbury advanced fast Fourier Transform (DaFT) routine which is fully memory and domain distributed 3D FFT routine used for calculation of long ranged forces in DL_POLY_3. Andrew Porter is the developer of NetCDF I/O routines in DL_POLY_3.

References:

1. *The DL_POLY webpage*, http://www.ccp5.ac.uk/DL_POLY/.
2. I.T. Todorov and W. Smith, 2004, *Phil. Trans. R Soc. Lond.*, **A 362**, 1835.
3. M.R.S. Pinches, D. Tildesley, and W. Smith, 1991, *Mol. Simulation*, **6**, 51.
4. I.T. Todorov, W. Smith, K. Trachenko and M.T. Dove, *J. Mater. Chem.*, **16**, 1611.
5. I.T. Todorov, N.L. Allan, J.A. Purton, M.T. Dove and W. Smith, *J. Mater. Sc.*, **42** (6), 1920.
6. *Report on the Jülich Blue Gene/L Scaling Workshop 2006*, FZJ-ZAM-IB-2007-02, http://www.fz-juelich.de/jsc/docs/printable/ib/ib-07/ib-2007-02.pdf.
7. *The HPCx Supercomputing Facility*, http://www.hpcx.ac.uk/.
8. *The Science and Technology Facilities Council*, http://www.stfc.ac.uk/.
9. *The IBM BlueGene*, http://www.research.ibm.com/bluegene/.
10. *HECToR - The UK Supercomputing Service*, http://www.hector.ac.uk/.
11. *The Swiss National Supercomputing Centre*, http://www-users.cscs.ch/.
12. *NetCDF*, http://www.unidata.ucar.edu/software/netcdf/.
13. K. Hinsen, *J Comp Chem*, **21**, 79.
14. AMBER NetCDF conventions http://amber.scripps.edu/netcdf/nctraj.html.