# DL_POLY_3: science on a grand scale via massively parallelised molecular dynamics simulations

IT Todorov, W Smith (CCLRC Daresbury Laboratory)

K Trachenko, MT Dove (University of Cambridge)

DL_POLY_3 [1] is a general purpose molecular dynamics (MD) package for simulation of large-scale systems. By inherent parallelism and full memory distribution, DL_POLY_3 excels in performance for large-scale systems on multi-processor clusters. The performance results from intricately interlaced numerical algorithms such as domain decomposition (DD), linked cells (LC) [2] and Daresbury advanced Fourier transform (DAFT) [3]. A reinforced numerical stability so important for highly non-equilibrium simulations, such as radiation damage (RD) cascades [4], is provided by symplectic integration based on Trotter derived velocity Verlet (VV) algorithms [5]. The current version (3.04) is written in suitably modularised FORTRAN90 with embedded MPI and is fully self-contained (no external libraries needed) which makes the code highly platform portable [6].

## Science example

DL_POLY_3 offers a wide variety of force-fields and virtually no limit to the system size (current limit of sizes up to 610 million particles) which enable the user to describe complex systems to high accuracy and simulate them in ever-growing realism. At present, it is the availability of HPC resources, high CPU count with extra disk space for output storage that limit the researcher in taking full advantage of the package capability. However, new HPC resources, such as HPCx [7], allow a number of scientists to carry out research on systems from a few hundred thousand to a few million particles [8]. DL_POLY_3 has been routinely used in RD research, where the demand of large size systems is driven by the high energies involved in RD cascades [4, 8-10]. The motivation for such studies lie in the growing need to immobilise radioactive nuclear waste safely by putting it in a waste form that can be an effective barrier to prevent polluting the environment. Safe immobilisation is therefore crucial to the future of the nuclear power industry. Even now, the amount of stored non-immobilised waste is already large enough to pose serious concerns, and is growing.

It is important to assess what consequences high-energy irradiation may have on the performance of the waste form over time, which can vary from about one hundred to tens of thousands of years for different isotopes. To study how properties of waste forms change over time massive parallel MD simulations of high-energy recoils are carried out in materials of interest. A recoil simulates a post alpha-decay event, ie after the alpha particle has transferred most of its energy and impulse to a heavy atom within the host matrix. This causes most of the damage in the structure, resulting in thousands of permanent atomic displacements. In our studies of RD effects [4,9,10] we consider materials related to those proposed to encapsulate highly radioactive nuclear waste (waste forms, glasses) as well as new ceramic materials indicating considerably higher durability. Figure 1 shows how marked the difference in damage (shape- and size-wise) can be for such different materials. These pictures correspond to the settled damage in $SiO_2$ and $TiO_2$ after 15 ps from a 50 keV recoil. To simulate this high energy recoil a 2.5 million particle simulation was carried out on 256 CPUs of HPCx running for 90 minutes per each compound.

Data from these sort of simulations on various materials are used to quantify a number of properties such as resistance/susceptibility to amorphisation, glass-forming ability, stopping power of irradiation, cascade-quenching, crystalline recoverability (self-healing), etc. These are used to improve scientific understanding

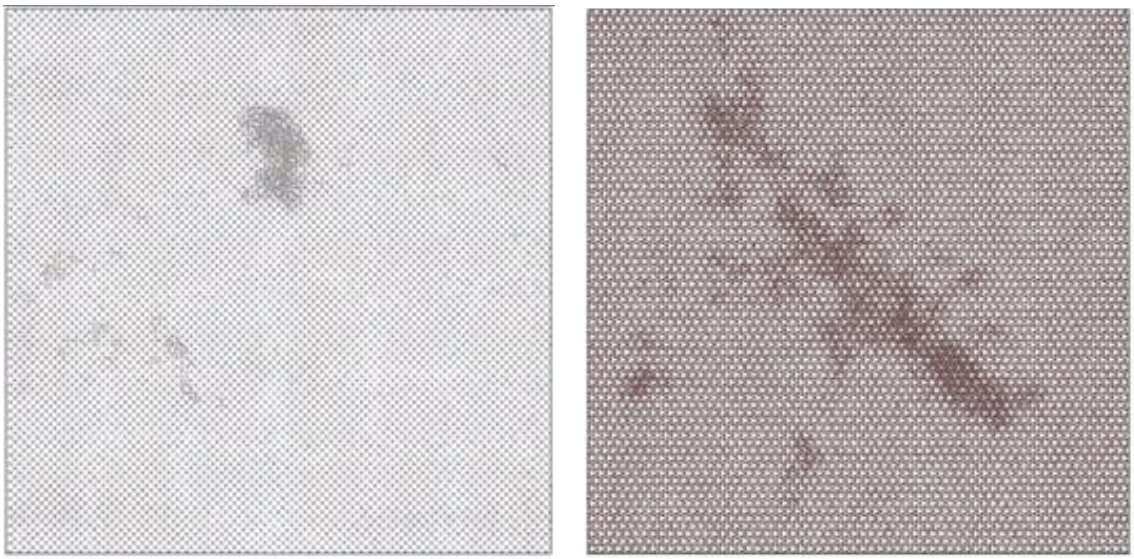


Figure 1: Damage in $TiO_2$ (left) and $SiO_2$ (right) after 15 ps from a 50 keV recoil. The simulated system size is 2.5 million particles.

Each simulation was carried out on 256 CPUs of HPCx running for 90 minutes.

and build theories of how the mechanisms of resistance to damage and healing from damage are related to the chemical nature of the material and eventually to suggest better candidates for waste forms [10].

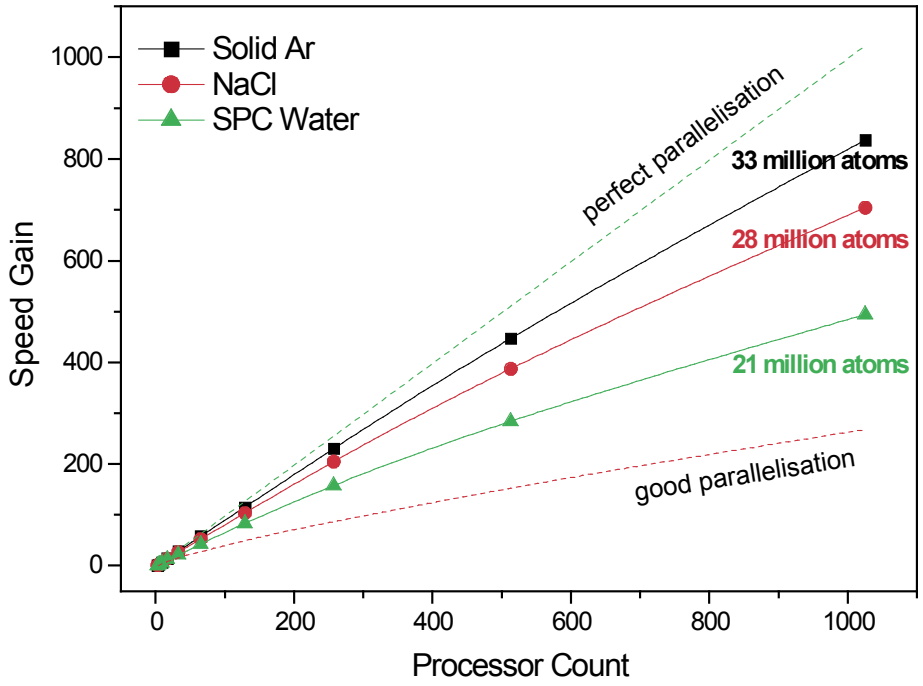



Figure 2. DL_POLY_3 speed gain plotted as a function of processor count. The green dotted line indicates the parallelisation limit also called perfect or embarrassing parallelisation. The red line indicates the standard for a good parallelisation, speed gain = $1.75^{\log_2(\text{CPU count})}$.

## Benchmarking performance

To investigate DL_POLY_3 performance on HPCx we carried out simulations on three model systems as listed in Table 1.

The quality of parallelisation was benchmarked over a large processor count by comparing simulation times per integration timestep as the ratio system size – processor count was kept constant. In such circumstances perfect parallelism would result in unchanged simulation times at any processor count and a speed-gain equal to the CPU count. In Figure 2 the speed gain is plotted as a function of CPU count for the three model systems. Also included in the figure are plots indicating perfect (linear) and good parallelisation. This plot clearly shows that parallelisation performance for each model system is better than the nominal for good parallelisation at any processor count.

The deviation from perfect parallelisation in the case of the Solid Ar system is due to inevitable communication organisation losses which increase with the processor count. The deviation in parallelisation performance between the NaCl and the Solid Ar systems results from the NlogN scalability of the DAFT algorithm with system size (N), which is used for calculation of the long-ranged Coulomb interactions (as part of the NaCl system force-field). The deviation in performance between the SPC Water and NaCl systems comes from the further complexity in the force-field of the former system, which includes constraint bonds. Constraints are handled by the iterative algorithms, RATTLE [11] or SHAKE [13] which involve extra communications at each iteration. Overall, DL_POLY_3 exhibits excellent parallelisation over a wide range of CPU counts for which it has been awarded a Gold Star on HPCx. It is also worth noting that DL_POLY_3 holds the current record for the largest simulation (28 million ions on 1024 CPUs for the NaCl system) by a general purpose MD code including full evaluation of the Coulombic interactions.

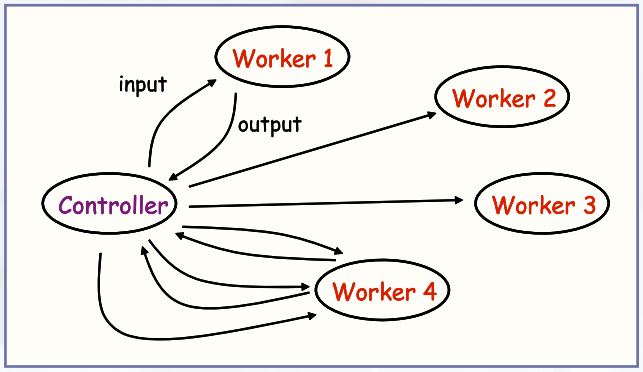
Maximum load test involving the three basics systems were also run on HPCx. We increased the MD simulation size load on one CPU (≈ 0.8 GB memory) incrementally until execution failed. The maximum load for the Solid Ar system was 600 000 particles per CPU. This on 1024 CPUs would add up to ≈ 610 million particles, which is just below the limit DL_POLY_3 can handle before some integer accumulators exceed their 32-bit declaration limit. With a little work this limit can be changed to the real 32-bit limit (2 147 483 647). Even this limit can be exceeded with careful effort and system sizes of $9.2\times10^{18}$ are possible. To achieve this the package must be compiled in 64-bit mode and executed on a 64-bit platform requiring considerable hard disk space. For a 2.5 million particle system a configuration file (lattice parameters, positions, velocity and forces) in textual format is ≈ 1 GB.

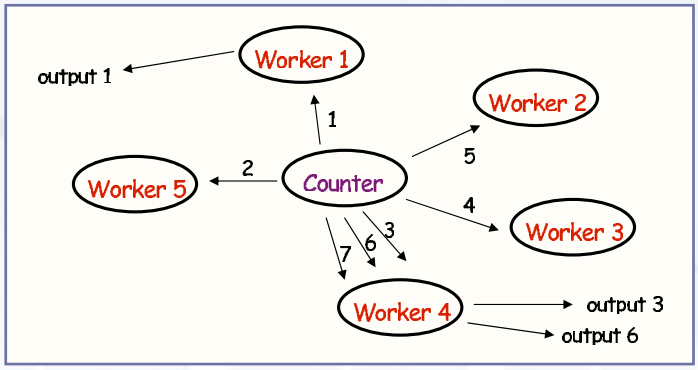
The maximum load per CPU for the NaCl and SPC Water systems amounted to 180 000 and 170 000 ions respectively. These cannot be scaled so directly to 1024 CPUs since some extra memory per CPU will be needed for the 3D FFTs in the SPME summations[13]

*Continued on page 14.*

Table 1. The model systems simulated using DL_POLY_3.04.

| System | Size per CPU [particles] | Ensemble [type] | Force-field complexity | Cutoff [Å] | Temperature [K] | Pressure [k bar] |
|---|---|---|---|---|---|---|
| Solid Ar | 32 000 | NVE | short range | 9 | 4.2 | 0.001 |
| NaCl | 27 000 | NVE | above & Coulomb | 12 | 500 | 0.001 |
| SPC Water | 20 736 | NPT Berendsen 0.5 0.75 | above & constraints | 8 | 300 | 0.001 |




Left: A classical task farm with a separate controller process. Above: a task farm comprising only workers accessing a shared counter.

# Task farming on HPCx

David Henty

Although HPCx is primarily intended to run single parallel applications that scale to many hundreds of processors, there are situations where it can be beneficial to run simultaneously a large number of smaller jobs which are loosely coupled. These small jobs could be parallel applications that perhaps do not scale well, or even sequential programs. Typical examples include statistical sampling, ensemble modelling, parameter searches and pre- or post-processing of data.

The way that the parallel environment is implemented on IBM systems means that there is a very tight coupling between LoadLeveler and the 'poe' command (the parallel job launcher). The effect is that it may not immediately be obvious how to do anything other than launch a single large MPI job across all the requested CPUs.

There have been quite a number of requests to the HPCx Helpdesk for assistance with taskfarming, for example:

- Is it possible to run multiple serial jobs?
- Will these jobs be distributed across the whole machine or will they all run on a single node?
- How can I distinguish between jobs so that they can perform different tasks?

To help address these issues, I have written a simple harness called 'taskfarm' that lives in /usr/local/packages/bin/. This allows users to run multiple serial jobs in a straightforward way across any number of processors, and gives each of them access to a unique identifier analogous to the MPI rank. For details, see the FAQ entry 'How can I run a task farm on HPCx ...' at http://www.hpcx.ac.uk/support/FAQ/.

It is somewhat more difficult to run multiple parallel jobs at the same time. This can still be achieved using 'taskfarm' if the jobs are not parallelised with MPI, eg if they use another mechanism like OpenMP or explicit threads. Running multiple MPI jobs, however, requires changes to the source code, but in some cases this can be relatively straightforward. For a general discussion see the talk 'Task Farming on HPCx', which was presented at the HPCx User Group meeting in July and is linked from the FAQ entry mentioned above.

## *DL_POLY_3 continued*

(used for evaluating the Coulomb interactions) if the SPME precision is kept constant. We estimate that we should be able to load such systems on 1024 CPUs to 1000 times the maximum load per CPU.

### References:

[1] Todorov I.T. and Smith W., R. Soc. Phil. Trans.: MPES, Theme DEM issue, 362, 1835 (2004)

[2] Pinches M.R.S., Tildesley D., Smith W, Mol. Simulation., 6, 51 (1991)

[3] Bush I.J., The Daresbury Advanced Fourier Transform, Daresbury Laboratory 1999

[4] Trachenko K., Dove M.T., Geisler T., Todorov I.T. and Smith W., J. Phys.: Cond. Matt., 16 (27), S2623, (2004)

[5] Martyna G.J., Tuckerman M.E., Douglas J.T., Klein M.L., Mol. Phys., 87 (5), 1117 (1996)

[6] Todorov I.T., Smith W., 'DL_POLY_3: new dimensions in Molecular Damage Simulations via Massive Parallelism', J. Mat. Chem., (in progress)

[7] IBM SP4 cluster – http://www.hpcx.ac.uk;

[8] Trachenko K., Dove M.T., Pruneda M., Artacho E., Salje E., Geisler T., Todorov I.T. and Smith W., MRS invited paper, Mat. Res. Soc. Symp. Proc. 792, R6.2.1 (2004)

[9] Trachenko K., Dove M.T., Ekhard K.H. Salje E., Todorov I.T., Smith W., Pruneda M. and Artacho E., Mol. Sim., 31 (5), (2005)

[10] Trachenko K., J. Phys.: Cond. Matt., 16, R1491–R1515 (2004)

[11] Andersen H.C., J. Comput. Phys., 52, 24 (1983)

[12] Ryckaert J.P., Ciccotti G., Berendsen H.J.C., J. Comp. Phys., 23, 327 (1977)

[13] Essmann U., Perera L., Berkowtz M.L., Darden T., Lee H., Pedersen L.G., J. Chem. Phys., 103, 8577 (1995)