Implementation of Dual Resolution Simulation Methodology in LAMMPS

Iain Bethune[1],

Sophia Wheeler[2], Samuel Genheden[2,3],

and Jonathan W. Essex[2]

[1]EPCC, The University of Edinburgh

[2]Department of Chemistry, The University of Southampton

[3]University of Gothenburg

Version 1.2, October 19, 2016

1 Overview

This report documents the achievements of ARCHER eCSE project 04-07. Six months of effort were funded between August 2015 and July 2016, and the work was carried out by Iain Bethune at EPCC.

The overall aim of the project was to enable simulations using the ELBA force-field [1] to be carried out within the LAMMPS [2, 3] program. ELBA is an ELectostatics-BAsed coarse-grained force-field for modelling lipid systems which is unique in that simulations consist of both spherical beads which may be charged or dipolar, and classical atomistic particles. In this “dual resolution” model, the lipid and water environment is modelled relatively cheaply with the coarse-grained beads (as shown in Figure 1) while solute molecules with important chemistry are modelled at full atomistic detail. Particles interact via bonded (bond, angle, dihedral) terms as well as explicit charge-charge, charge-dipole and dipole-dipole non-bonded electrostatics.

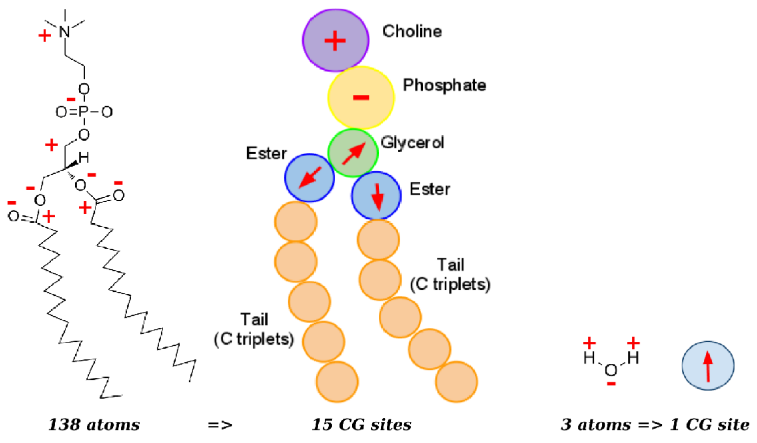


Figure 1: ELBA coarse-grained model of a DSPC phospholipid and water molecule, showing the use of charged, neutral and dipolar beads. Reproduced from [4].

The ELBA force-field was first implemented in the BRAHMS code [5], developed in the Essex group. The main limitation of BRAHMS is that the code is serial, limiting the size of system which can be realistically studied, as well as having only a small user base. By contrast, LAMMPS has a large user community, demonstrated scalability, and an extensible structure. At the outset of the project the particle pair interactions which comprise the ELBA force-field are implemented, as well as a multiple-timestepping r-RESPA [6] algorithm to allow the atomistic system to be propagated at a shorter timestep than the coarse-grained part, increasing the simulation speed. In order to tackle systems of interest such as peptides and proteins embedded in lipid membranes three further features must be implemented in LAMMPS:

- Implementation of a symplectic and time-reversible rigid body integrator to allow stable long-duration MD.
- Modification of the existing Parrinello-Rahman barostat to support the new integrator.
- Optimisation and improved load balancing for improved efficiency of parallel dual resolution simulations.

These features have been implemented in a fork of the main LAMMPS development code. This code is available from http://www.github.com/ibethune/lammps and are installed on ARCHER as the module `lammps/elba`. The code has been developed based on the `lammps-icms` branch to aid eventual inclusion into a future LAMMPS official release. The new integrators have been incorporated into the latest stable release (30 July 2016) of LAMMPS. The load balancing functionality has been merged into `lammps-icms`, and should appear in a future release.

2 Rigid Body Integration

LAMMPS already includes several 'fixes' (classes which implement the integration step for a particular class of particles) which are relevant for rigid bodies. The simplest of these is `fix nve/sphere`, which integrates the position and velocity of spherical bodies using the Velocity Verlet scheme, as well as the angular momentum. If the particle has a defined dipole and the `update dipole` option is specified, the orientation of the dipole is also integrated, using the update (subject to normalisation):

$$\boldsymbol{\mu}_{t+\delta t} = \boldsymbol{\mu}_t + \delta t \cdot \boldsymbol{\omega} \times \boldsymbol{\mu}_t$$

Where $\boldsymbol{\mu}$ is the dipole vector and $\boldsymbol{\omega}$ the angular velocity. This integrator was found not to correctly conserve rotational kinetic energy (see Figure 2, for example), so we have implemented the integrator of Dullweber, Leimkuhler and McLachlan [7] (henceforth referred to as DLM). The DLM algorithm is proven to be symplectic and time-reversible, thus giving good energy conservation and stability for long simulations. The canonical DLM implementation stores the particle orientation as a 3x3 rotation matrix $\boldsymbol{Q}$. In ELBA the orientation of the beads is defined only by the dipole, as they are otherwise rotationally symmetric. The LAMMPS `atom_style` used in the simulations already stores the dipole, so rather than adding an additional property which would require additional storage (and communication), we take the dipole to define the z-axis of the body-frame and construct the rotation matrix on-the-fly as follows. Defining the rotation matrix $\boldsymbol{Q}$ as the rotation which takes a vector from the space-frame to the body-frame (as indicated by the vector subscripts),

$$\boldsymbol{v}_b = \boldsymbol{Q} \cdot \boldsymbol{v}_s$$

We define (closely following the algorithm of [8]):

$$\boldsymbol{a} = \boldsymbol{\mu}/||\boldsymbol{\mu}||$$

$$\boldsymbol{v} = \boldsymbol{a} \times [001]$$

$$s = ||\boldsymbol{v}||$$

$$c = \boldsymbol{a} \cdot [001]$$

$$[v]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

Then the rotation matrix $\boldsymbol{Q}$ is given by:

$$\boldsymbol{Q} = \boldsymbol{I} + [v]_\times + [v]_\times^2 \frac{1-c}{s^2}$$

Care has to be taken for the case were the dipole is parallel (or anti-parallel) to the space-frame z-axis, in which case $\boldsymbol{Q} = \boldsymbol{I}$ (or $-\boldsymbol{I}$, respectively). In the DLM algorithm the orientation is integrated by a sequence of rotations around each axis *in the body-frame*, while LAMMPS stores the angular velocity in the space-frame. Thus compared to the steps given in the DLM paper, our implementation has an additional transformation step to compute the angular velocity in the body-frame, and due to the definition of the orientation matrix, the orientation updates take the form of a right-multiplication by the transpose of the rotation matrix. The rotation matrices for a rotation around each axis are defined as (x rotation shown, the other two axes similarly):

$$\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

and implemented (in the `MathExtra` namespace), using the more efficient small angle approximations:

$$\sin\phi = \frac{\phi}{1+\phi^2/4}, \qquad \cos\phi = \frac{1-\phi^2/4}{1+\phi^2/4}$$

The full rotational update is implemented as follows:

$$\begin{aligned} &\omega_b = \boldsymbol{Q}\omega_s \\ &\boldsymbol{R}_1 = \boldsymbol{R}_x(\frac{\delta t}{2}\omega_1), \quad \omega = \boldsymbol{R}_1\omega, \quad \boldsymbol{Q} = \boldsymbol{R}_1^T\boldsymbol{Q} \\ &\boldsymbol{R}_2 = \boldsymbol{R}_y(\frac{\delta t}{2}\omega_2), \quad \omega = \boldsymbol{R}_2\omega, \quad \boldsymbol{Q} = \boldsymbol{R}_2^T\boldsymbol{Q} \\ &\boldsymbol{R}_3 = \boldsymbol{R}_z(\delta t\omega_3), \quad \omega = \boldsymbol{R}_3\omega, \quad \boldsymbol{Q} = \boldsymbol{R}_3^T\boldsymbol{Q} \end{aligned}$$

$$\boldsymbol{R}_4 = \boldsymbol{R}_y(\frac{\delta t}{2}\omega_2), \quad \boldsymbol{\omega} = \boldsymbol{R}_4\boldsymbol{\omega}, \quad \boldsymbol{Q} = \boldsymbol{R}_4^T\boldsymbol{Q}$$

$$\boldsymbol{R}_5 = \boldsymbol{R}_x(\frac{\delta t}{2}\omega_1), \quad \boldsymbol{\omega} = \boldsymbol{R}_5\boldsymbol{\omega}, \quad \boldsymbol{Q} = \boldsymbol{R}_5^T\boldsymbol{Q}$$

Finally, the space-frame dipole is computed as:

$$\boldsymbol{\mu}_s = \boldsymbol{Q}^T[001] \cdot ||\boldsymbol{\mu}||$$

To correctness of the implementation was tested on two systems. Firstly, a 49.34 Å^3 cubic box containing 4000 ELBA water beads was equilibriated at 300K using a Langevin thermostat for 20 ps, then the thermostat was removed and 20 ps of MD in the microcanonical (NVE) ensemble was performed. The timestep for both the entire run was 10fs using `fix nve/sphere update dipole`. Two runs, one with the native 'LAMMPS' integrator and one with our new 'DLM' integrator are shown in Figure 2. While the thermostat is enabled equipartition of the kinetic energy is maintained in both cases. Once the thermostat is removed, the 'leaking' of energy from the rotational modes is clear, and good energy conservation is only obtained by using the DLM integrator.

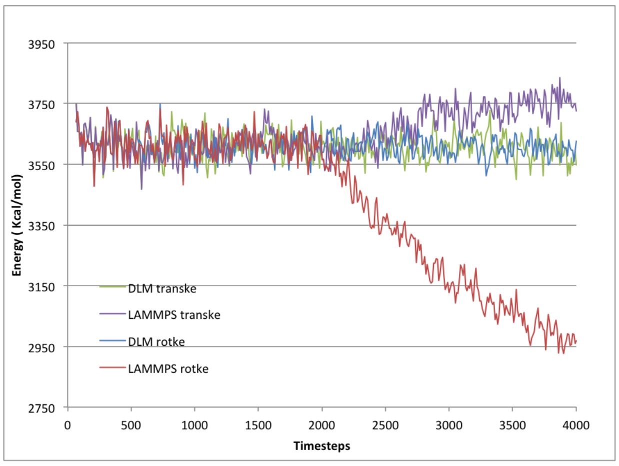


Figure 2: Rotational and translational components of kinetic energy for a box of 4000 ELBA water beads during 2000 steps of NVT and 2000 steps of NVE molecular dynamics.

A similar experiment was carried out with a simulation of a lipid membrane consisting of 128 DMPC molecules in water, running over a total of 175 ps. The first 75ps are NVT (using a Langevin thermostat), and the remaining 100ps are NVE. Firstly, as shown in Figure 3, the total energy is conserved during NVE MD only when the DLM integrator is used. Secondly, the integrator is shown to be stable for timesteps up to 16fs (Figure 4), so is well suited to use in production simulations. To enable the new integrator simply add the keyword/value pair `update dipole/dlm` to the `fix nve/sphere` command.

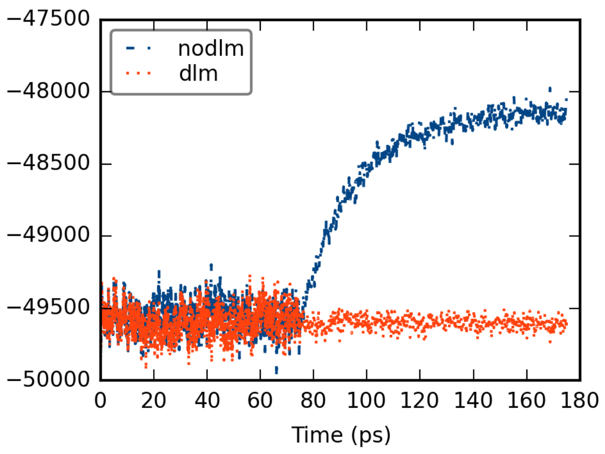


Figure 3: Total energy for a lipid membrane system, showing the excellent energy conservation of the DLM algorithm with a 10fs timestep.

3 Constant Pressure Molecular Dynamics

The `fix nve/sphere` has several limitations. By itself it does not have any temperature control, although it may be combined with a separate thermostat such as `fix langevin` or `fix temp`. For accurate modelling of the lipid membrane environment, and to allow exploration of phenomena such as phase change the ability for the simulation box to deform in response to internal stresses, subject to a fixed external pressure, is needed. Combined with a thermostat, this describes the *isobaric-isothermal* or NPT ensemble. In LAMMPS there are a set of related fixes which integrate spherical particles in ensembles where the temperature or pressure or both are controlled by Nosé-Hoover [9]:

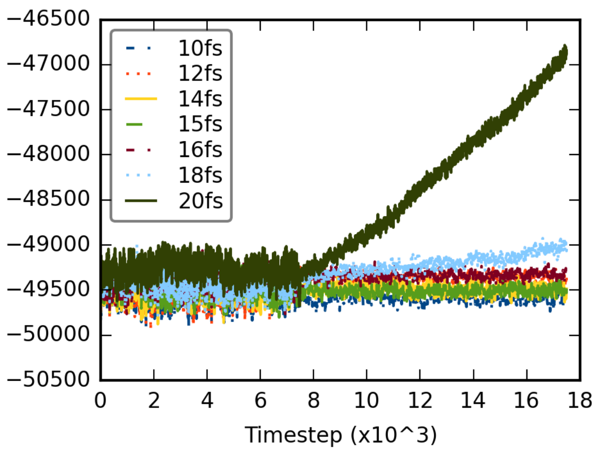


Figure 4: Total energy as a function of the timestep for a lipid membrane system.

- `fix nvt/sphere`: A Nose-Hoover thermostat is applied to the translation and rotational degrees of freedom, producing the canonical (NVT) ensemble.
- `fix npt/sphere`: A Nose-Hoover thermostat is applied to the translation and rotational degrees of freedom and a Nose-Hoover barostat (also known as the Parrinello-Rahman barostat [10]) is applied to the the simulation box, producing the isobaric-isothermal (NPT) ensemble.
- `fix nph/sphere`: A Nose-Hoover barostat is applied to the simulation box, producing the isenthalpic (NPH) ensemble.

Each of these fixes is implemented by a class which inherits from `FixNHSphere`. We have implemented the DLM integrator (as described in Section 2) in this base class, so the functionality is available in all three subclass fixes. As per `fix nve/sphere`, the DLM integrator is enabled by the keyword/value pair `update dipole/dlm` to the `fix` command. To test the correctness of the implementation (although the exactly the same code is used as for `fix nve/sphere` we ran 10,000 steps of NPT MD with a timestep of 10fs. Due to the thermostat and barostat, the native integrator does not exhibit the same drift shown in Figure 2. However, as shown in Figure 5 after a short relaxation period the translational and rotational KE are conserved (and the rotational energy is 2/3 of the kinetic energy as expected due to having one less degree of freedom since they are symmetric under rotation about the dipole axis). The pressure also relaxes quickly, displaying short-term fluctuations typical for such as small and compressible system.

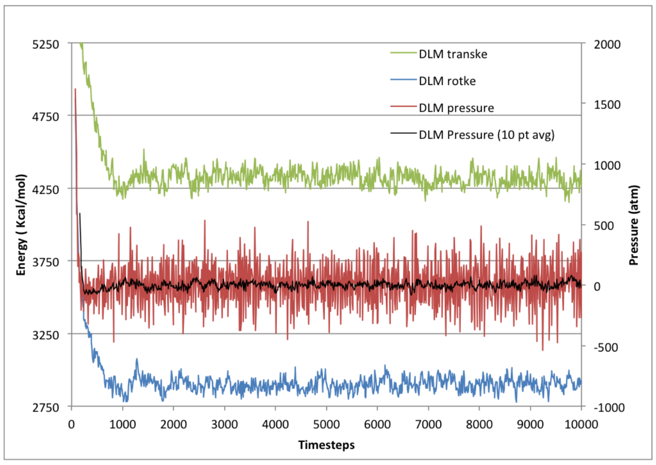


Figure 5: Components of kinetic energy and computed pressure for a box of 4000 ELBA water beads during 10000 steps of NPT molecular dynamics.

4 Performance optimisation and load balancing

For performance testing, we have used a dual resolution model of the BPTI protein solvated in water. Bovine pancreatic trypsin inhibitor (BPTI) is a single-chain polypeptide which has been widely studied with MD (for example, in [11] a 1 ms atomistic MD trajectory was generated by the Anton computer). It contains 58 amino acid residues, here represented by 882 atoms, and is solvated in a 60.22 Å× 55.97 Å× 56.87 Åbox, containing 6136 ELBA water beads, giving a total of 7018 particles. Integration is performed by `fix nve/sphere update dipole/dlm`, `fix langevin` thermostats the system, and a `fix press/berendsen` barostat is added to produce the NPT ensemble.

4.1 Serial Optimisation

In the ELBA force-field, in addition to bonded terms, inter-particle forces come from two non-bonded pairwise interaction types. For pairs involving ELBA beads, `pair_style lj/sf/dipole/df` (from the `USER-MISC` package) which includes Lennard-Jones and dipole-dipole, dipole-charge and charge-charge coulombic terms up to the defined cutoff radius. The a shifted potential is employed so the energy and force go to zero at the cut-off. For all other (atomistic) interactions the standard CHARMM L-J and coulomb terms (`pair_style  lj/charmm/coul/long`, see [12]) are used. The long range electrostatics are handled by one of the LAMMPS kspace solvers such as `pppm/cg`.

When running on a relatively small number of MPI processes (or in serial), the pair-force computation dominates, taking over 80% of the total runtime. Profiling with Cray-PAT showed that most of this was from the functions `LAMMPS_NS::PairLJCharmmCoulLong::-compute` and `LAMMPS_NS::PairLJSFDipoleSF::compute`, i.e. the functions that computes the two interaction types in the ELBA force-field. Analysis of the code revealed several opportunities for optimisation, in particular pre-computing reciprocals to avoid expensive floating-point divisions at each MD step. For the BPTI simulation, this resulted in a 4% speedup, as shown in Figure 6.

4.2 Load balancing

LAMMPS employs a spatial domain decomposition where the simulation box is partitioned into regions, and each region is assigned to an MPI process. That process then ‘owns’ any particles within its own region, and is responsible for applying fixes to those particles, computing pair forces on those particles, and contributing to the parallel kspace solver. LAMMPS supports non-orthorhombic boxes, but all load balancing is done in ‘lambda coordinates’ where positions in space are represented as fractions of the three lattice vectors. Thus, for ease of explanation we will use orthorhombic boxes in 2D, although all of the following generalises to 3D. By default, the number of MPI processes is factorised into 3 dimensions ($NP = NP_x \times NP_y \times NP_z$), and the box is divided evenly in each dimension, so each process has the same sized domain. While straightforward, this may result in load imbalance if the number of particles (or the amount of work associated with the particles) varies from one domain to another. For dual resolution simulations with ELBA this is exacerbated by three factors. Firstly, ELBA beads occupy more volume than atomistic particles, so they are expected to be less densely distributed in space. Secondly, because each `pair_style` has a different functional form the amount of time taken per pair will vary depending on the type of particle (bead or atomistic, dipolar and/or charged). Thirdly, the dual resolution formulation of ELBA naturally lends itself to a multiple timestepping approach (implemented by `run_style respa`), which also has implications for load balancing. In r-RESPA, a set of nested timesteps are defined - usually two, but more are possible in principle. Particular interactions (and optionally, fixes) can be computed at different timesteps. Usually, the most expensive (but slowly varying) interactions will be computed at the outer timestep, thus saving computational effort. Rapidly varying forces must be computed at the inner timestep. For example, the BPTI simulation uses the following r-RESPA settings:

```
pair_style hybrid lj/sf/dipole/sf 12.0 \
   lj/sf/dipole/sf 12.0 lj/charmm/coul/long 11.0 12.0
...
timestep 8.0
run_style respa 2 2 hybrid 1 2 1 kspace 1 improper 2
```

This defines an outer timestep of 8 fs and 2 r-RESPA levels with a timestep ratio of 2 so that the inner timestep is 4 fs. The remaining options set at which level each type of forces are computed: here the first and third parts of the `hybrid` pairs are computed at level 1 (the inner timestep), corresponding to pair forces involving atomistic particles, and the second part (water-water pairs) are only computed at the outer step. Likewise, the long-range electrostatic forces (`kspace`) are computed at the level 1, but the forces due to improper dihedral terms at level 2. By default, all of the intra-molecular forces (bonds, angles, dihedrals...) are computed at the innermost level. This causes further load imbalance, since in this case, the pair and bonded forces on the solute are evaluated twice as often as the water-water pairs. This effect is exacerbated when the timestep ratio is increased.

Two different load balancing methods are available in LAMMPS: `shift` and `rcb`, illustrated in Figure 7. Load balancing may be static (once at the beginning of the run) using `balance`, or dynamic (rebalancing after at a fixed frequency during the run), using the `fix balance` command. `shift` works on each dimension in turn, and adjusts (or shifts) the divisions in that dimension so each section has the same total number of particles. Depending on the options supplied to `balance` the dimensions may be balanced in various orders, or not at all. This method may not obtain a perfect load balancing of particles, but does maintain the simple regular grid communication structure where each process has only 26 nearest neighbours (domains which share a face, edge, or corner). `rcb` implements a Recursive Coordinate Bisection. Iterating over dimensions, each is cut into two sub-domains, each with the same number of particles. Each sub-domain is then partitioned into two further sub-domains, and this is applied recursively until there are enough domains for all the processes. Clearly, the RCB by construction will result in a decomposition which is perfectly load balanced. However, the more complex decomposition structure means that each domain may have an arbitrary number of neighbours, and so the more flexible `comm_style tiled` communication scheme must be used, which in turn places limitations on which `kspace` solvers may be used. In particular, with `balance rcb` and `comm_style tiled`, `kspace_style pppm/cg` is not supported and the (slower) `kspace_style ewald` must be used instead. Thus, even though `rcb` offers a potentially better load balance, in practice we found it to be slower than using `shift` due to the extra time spent in the kspace solver.

As discussed earlier, the main problem with both load balancing approaches is that they assume that the amount of work (the load) is proportional to the number of particles. For dual-resolution simulations (or indeed any simulation where the particles are not homogeneous) this is clearly not the case, as shown in Figure 8. Despite the RCB balancing the number of particles to within 0.2% of a perfect load balance, the amount of time is still imbalanced by over 100% for the most significant parts of the calculation!

To address this we introduce the concept of a load factor - instead of treating every particle as equal i.e. 1 unit of load, we allow particles to have arbitrary load associated with them, so they have a higher weighting in the load balancing algorithm. For example,

```
Loop time of 120.903 on 1 procs for 1000 steps with 7018 atoms

Performance: 5.717 ns/day, 4.198 hours/ns, 8.271 timesteps/s
99.3% CPU use with 1 MPI tasks x no OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 103.35     | 103.35     | 103.35     |   0.0 | 85.48
Bond    | 1.095      | 1.095      | 1.095      |   0.0 |  0.91
Kspace  | 4.3415     | 4.3415     | 4.3415     |   0.0 |  3.59
Neigh   | 7.2393     | 7.2393     | 7.2393     |   0.0 |  5.99
Comm    | 1.1778     | 1.1778     | 1.1778     |   0.0 |  0.97
Output  | 0.0020359  | 0.0020359  | 0.0020359  |   0.0 |  0.00
Modify  | 2.9979     | 2.9979     | 2.9979     |   0.0 |  2.48
Other   |            | 0.6963     |            |       |  0.58
```

(a) Before optimisation

```
Loop time of 116.700 on 1 procs for 1000 steps with 7018 atoms

Performance: 5.923 ns/day, 4.052 hours/ns, 8.569 timesteps/s
99.4% CPU use with 1 MPI tasks x no OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 99.448     | 99.448     | 99.448     |   nan | 85.22
Bond    | 1.0763     | 1.0763     | 1.0763     |   0.0 |  0.92
Kspace  | 4.0912     | 4.0912     | 4.0912     |   0.0 |  3.51
Neigh   | 7.3013     | 7.3013     | 7.3013     |   0.0 |  6.26
Comm    | 1.1464     | 1.1464     | 1.1464     |   0.0 |  0.98
Output  | 0.00031209 | 0.00031209 | 0.00031209 |   0.0 |  0.00
Modify  | 2.9559     | 2.9559     | 2.9559     |   0.0 |  2.53
Other   |            | 0.6776     |            |       |  0.58
```

(b) After optimisation

Figure 6: LAMMPS timing reports for 1000 steps of NPT molecular dynamics of BPTI on a single process before and after pair force optimisations.

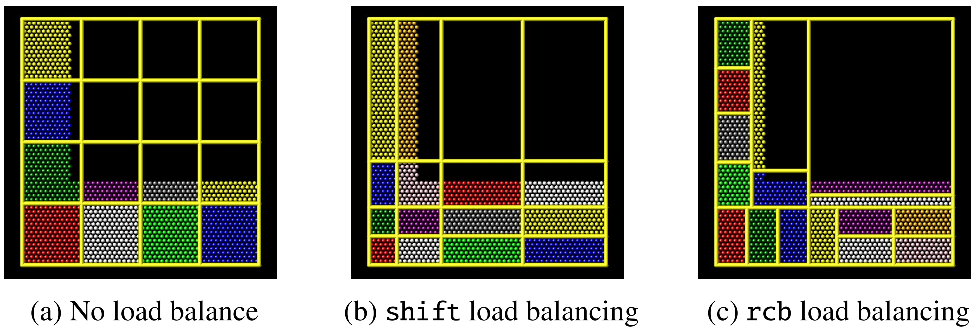
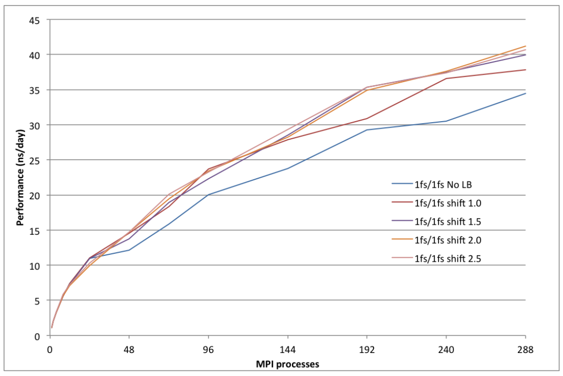


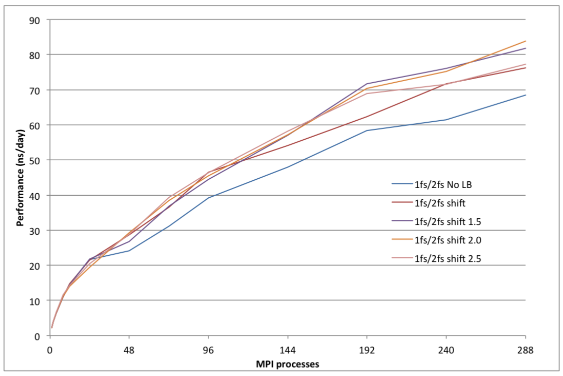
Figure 7: Examples of possible decompositions (in 2D) for a spatially inhomogeneous system. Reproduced from [13].

```
initial/final max load/proc = 455 293
initial/final imbalance factor = 1.556 1.00199
...
Loop time of 88.8132 on 24 procs for 5000 steps with 7018 atoms

Performance: 38.913 ns/day, 0.617 hours/ns, 56.298 timesteps/s
99.6% CPU use with 24 MPI tasks x no OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 17.062     | 21.717     | 33.217     | 103.3 | 24.45
Bond    | 0.01209    | 0.22359    | 0.73098    |  53.0 |  0.25
Kspace  | 37.053     | 52.369     | 60.539     | 110.4 | 58.97
Neigh   | 2.2107     | 2.2414     | 2.2719     |   1.0 |  2.52
Comm    | 4.1477     | 9.7616     | 14.753     | 105.8 | 10.99
Output  | 0.00089192 | 0.0010348  | 0.0032897  |   1.5 |  0.00
Modify  | 2.063      | 2.2138     | 2.353      |   5.2 |  2.49
Other   |            | 0.2852     |            |       |  0.32
```

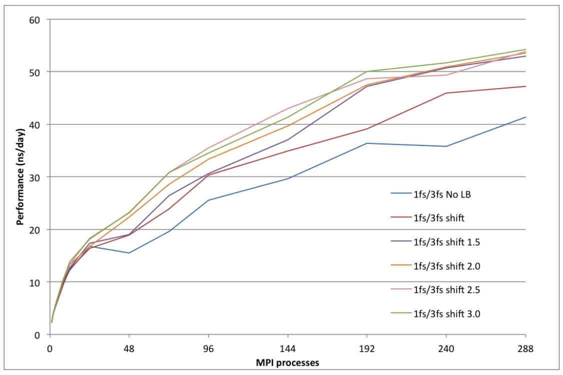
Figure 8: LAMMPS particle and measured load imbalance for BPTI with r-RESPA timestep ratio of 2, on 24 MPI processes. Load balancing is performed once at the beginning of the run.



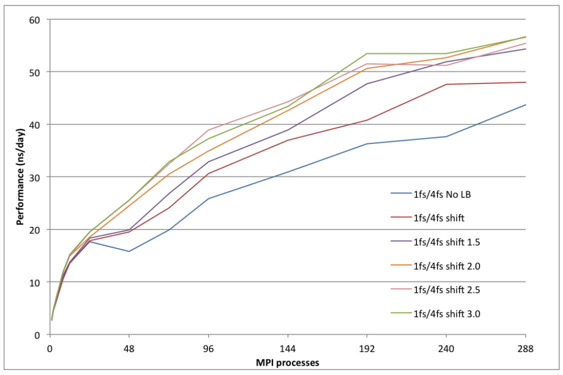
(a) 1fs/1fs



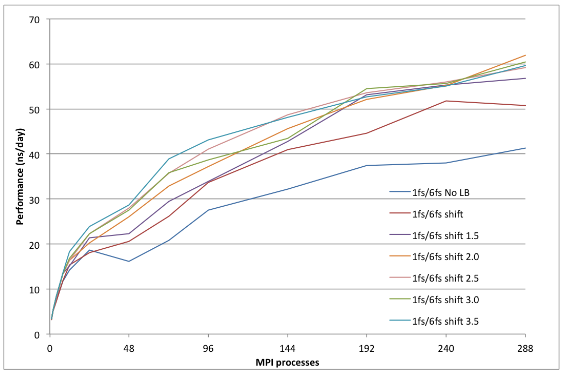
(b) 1fs/2fs



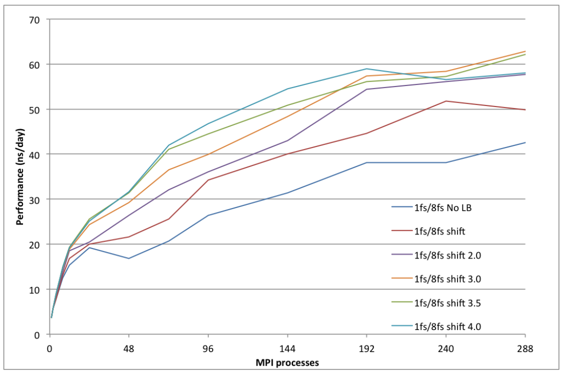
(c) 1fs/3fs



(d) 1fs/4fs



(e) 1fs/6fs



(f) 1fs/8fs

Figure 9: Scaling behaviour of LAMMPS for the BPTI system with and without load balancing, and various settings for the solute load factor.

a particle with a load factor of 10.0 would balance 10 'normal' particles. LAMMPS already has a very flexible concept of a `group` which can be used to select particles with given attributes. The `balance` command has been extended with a `weight group` style which assigns the specified load to all particles in that group. For example `weight group 1 solute 2.0` applies a weight of 2.0 to all particles in the `solute` group. This setting is left to the user since it is difficult to predict *a priori*, the optimal value depending on the `atom_styles`, `pair_styles`, and `respa` settings. Since groups may overlap we assign a load factor of 1.0 to a group unless otherwise specified by the user, and the load factor of an individual particle is the product of the load factors of the groups of which it is a member:

$$particle_p \in \{group_0, ..., group_n\}$$

$$loadfactor_p = \prod_{i=0}^{n} loadfactor_{g_i}$$

In our approach, if the user does not specify any group load factors then the prior load balancing behaviour based on particle numbers is obtained.

In the BPTI system, we define two groups - the `solute`, containing of all the atomistic particles, and `water`, containing the ELBA water beads. Since the most of the forces on the `solute` particles are evaluated at the inner timestep, we expect that setting a load factor > 1 should improve the load balance, especially at high r-RESPA ratios. We ran performance tests to compare the scaling behaviour using no load balancing and load balancing with the `shift` method for a range of load factors and r-RESPA timestep ratios, and the results are shown in Figure 9. We tested only up to 288 MPI processes, although performance is still improving, even with load balancing the parallel efficiency is $< 15\%$ so not of practical interest to users.

Figure 9a shows that even when the r-RESPA ratio is 1 i.e. all forces are evaluated every step, the system benefits from load balancing with a modest load factor for the solute. Without any load balancing, the fact that the ELBA water beads are less dense than atomistic particles leads to domains containing only water having less load. However, it is possible to improve on the default load balance by setting a small `solute` load factor to account for the fact that the solute uses a different `pair_style` as well as having bonded interactions. As the r-RESPA timestep ratio increases, higher load factor values improve performance, as the amount of time spent evaluating forces on the solute particles increases proportionally. The speedups obtained for each timestep ratio on different processor counts are summarised in Table 1. While the optimum load factor setting varies depending on both the number of MPI processes and the r-RESPA ratio, it is clear that the weighted load balancing scheme outperforms the default in all cases. For moderate numbers of processes (where compute outweighs communications) a load factor of up to 4.0 gives best performance, while on highly parallel runs where communication costs are most significant, a smaller load factor of 2.0-3.0 is best. We recommend that users experiment to find the optimal values for their system.

r-RESPA	24 processes		72 processes		288 processes	
timesteps	Default	Best load factor	Default	Best load factor	Processes	Best load factor
1fs/1fs	0%	0% (1.5)	16%	27% (2.5)	10%	20% (2.0)
1fs/2fs	-1%	1% (1.5)	17%	26% (2.5)	11%	22% (2.0)
1fs/3fs	-3%	9% (3.0)	21%	57% (3.0)	14%	31% (3.0)
1fs/4fs	1%	11% (2.5)	21%	65% (3.0)	10%	30% (2.0)
1fs/6fs	-3%	28% (3.5)	26%	87% (3.5)	23%	50% (2.0)
1fs/8fs	4%	33% (3.5)	23%	103% (4.0)	17%	48% (3.0)

Table 1: Summary of speedups obtained for the BPTI system for a range of r-RESPA ratios and MPI processes.

5 Summary

At the conclusion of the project, we have implemented three new features in LAMMPS which make stable and efficient simulations using the dual resolution ELBA possible. From a user's perspective, the following new input keywords are available:

- `fix nve/sphere` takes the arguments `update dipole/dlm` to enable the DLM integrator.

- `fix nvt/sphere`, `fix npt/sphere` and `fix nph/sphere` take the argument `update dipole/dlm` to allow the DLM integrator to be combined with a range of thermostats and barostats.

- the `balance` command has an optional style `weight group Ngroup groupID-1 wight-1 groupID-2 weight-2 ...`, where `weight-1` is the weighting that should be applied to each particle in group `groupID-1` during load balancing etc. The weights are respected by both the `shift` and `rcb` load balancers. After merging with the `lammps-icms` branch, the implementation has been extended to cover dynamic load balancing (`fix balance`) and also several other weighting factors may be taken into account, including the number of neighbours, the time spent per process, or other computed `atom_style` properties. These have not been investigated within the scope of this project.

As they are fairly localised changes, the new DLM integration algorithm and the optimisations to the pair force compute functions have already been incorporated into the current stable release of LAMMPS (30 July 2016). The load balancing changes have been merged into the `lammps-icms` upstream repository for inclusion in the next public release. All of the functionality described in the report is available on ARCHER as a module `lammps/elba`.

Acknowledgements

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk). Oliver Henrich provided helpful review comments on an early draft.

References

[1] Mario Orsi and Jonathan W. Essex. The ELBA Force Field for Coarse-Grain Modeling of Lipid Membranes. *PLoS ONE*, 6(12):1–22, 12 2011. doi: 10.1371/journal.pone.0028637.

[2] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1 – 19, 1995. ISSN 0021-9991. doi: 10.1006/jcph.1995.1039.

[3] LAMMPS Molecular Dynamics Simulator, . URL http://lammps.sandia.gov.

[4] Sophia Wheeler. Simulating phase transitions in lipid bilayers using a coarse grained forcefield with molecular dynamics. 2013.

[5] BRAHMS-MD: Biomembrane Reduced-ApproacH Multiresolution Simulator for Molecular Dynamics. URL https://code.google.com/archive/p/brahms-md/.

[6] M. Tuckerman, B. J. Berne, and G. J. Martyna. Reversible multiple time scale molecular dynamics. *The Journal of Chemical Physics*, 97(3):1990–2001, 1992. doi: 10.1063/1.463137.

[7] Andreas Dullweber, Benedict Leimkuhler, and Robert McLachlan. Symplectic splitting methods for rigid body molecular dynamics. *The Journal of Chemical Physics*, 107(15):5840–5851, 1997. doi: 10.1063/1.474310.

[8] Jur van den Berg (http://math.stackexchange.com/users/91768/jur-van-den berg). Calculate rotation matrix to align vector a to vector b in 3d? Mathematics Stack Exchange. URL http://math.stackexchange.com/q/476311. (version: 2014-08-14).

[9] William G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31:1695–1697, Mar 1985. doi: 10.1103/PhysRevA.31.1695.

[10] M. Parrinello and A. Rahman. Crystal structure and pair potentials: A molecular-dynamics study. *Physical Review Letters*, 45(14):1196–1199, 1980. ISSN 00319007. doi: 10.1103/PhysRevLett.45.1196.

[11] David E. Shaw et al. Millisecond-scale Molecular Dynamics Simulations on Anton. pages 65:1–65:11, 2009. doi: 10.1145/1654059.1654126.

[12] AD MacKerell et al. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *The Journal of Physical Chemistry B*, 102(18): 35863616, April 1998. ISSN 1520-6106. doi: 10.1021/jp973084f.

[13] LAMMPS Docs - balance command, . URL http://lammps.sandia.gov/doc/balance.html.