Improving the performance of large-scale simulations in DL_MONTE

Tom L. Underwood, John A. Purton, Tina Düren, Stephen C. Parker

31/8/22

Abstract

DL_MONTE is a general-purpose program for simulating atomistic systems via the Monte Carlo (MC) method. We have made various improvements to this program to improve its capabilities with regards to *large-scale* simulations, i.e. simulations of systems comprised of large numbers of atoms or large molecules; or simulations which exploit a large number of parallel processes. To elaborate, we: 1) implemented the cell list method in DL_MONTE, to improve general performance of the program for large systems; 2) implemented a method, continuous fractional component MC, to enable insertion of molecules into a system during simulations – something routinely employed in MC studies of adsorption and fluid properties – in situations where standard approaches are intractable, e.g. solutions where the solvent and solute molecules have a large size difference; 3) made general improvements to the performance and scalability of the MPI parallelisation in DL_MONTE, both the parallelisation invoked in DL_MONTE to speed up energy calculations, and the parallelisation utilised by DL_MONTE to simulate multiple copies of the system in parallel via the replica exchange MC method. In this report we describe these improvements and evaluate the performance of DL_MONTE for large-scale simulations.

1 Introduction

Monte Carlo (MC) [1] is a powerful technique for studying condensed matter at the atomic scale. In MC the system is evolved stochastically to sample configurations at thermodynamic equilibrium. One pleasing feature of MC is that there is considerable freedom to tune the evolution of the system to suit the problem at hand; unlike molecular dynamics (MD) [1], the evolution need not be physical. For example, grand-canonical Monte Carlo (GCMC) is the method *par excellence* for studying adsorption; in GCMC the molecules are inserted into and removed from the system during a simulation [1]. MC can even utilise an extended system comprised of multiple coupled simulation boxes in order to solve a given problem; replica exchange Monte Carlo (REMC) [2], is a method for efficiently probing many temperatures, pressures or chemical potentials simultaneously in systems with rough energy landscapes, by simulating multiple copies of the system (i.e. *replicas*) in parallel.

While there are a plethora of open source general-purpose MD codes worldwide, there are only a few such MC codes. One of these is DL_MONTE [3,4,5]. DL_MONTE supports a wide range of MC methods, including GCMC and REMC. Moreover, it has a versatile force field, enabling it to simulate a wide range of systems including inorganic solids, fluids, and molecular crystals. It is also accompanied by extensive documentation and supporting software [5], including a Python library [6] which can be employed to build complex simulation workflows. (All releases and documentation can be obtained from the DL_MONTE GitLab page [5]).

A current limitation of existing MC codes is that, unlike well-established MD codes, they are not generally geared towards *large-scale* simulations, i.e. simulations of systems comprised of very large numbers of atoms, large molecules, complex force fields, or simulations which exploit large number of parallel processes [7]. This is unfortunate, because there are many problems which would benefit from a general-purpose MC code which can efficiently undertake such simulations.

DL_MONTE is well placed to become such a code. It was designed from the outset for HPC applications, supporting parallelisation via MPI to improve its performance. Moreover, it has been shown to perform favourably when compared to other general-purpose MC codes for 'realistic' force fields [7,8]. Motivated by this, we have recently undertaken work improve DL_MONTE's capabilities with regards to large-scale simulations. We describe this work here.

2 Cell lists

The first improvement to DL_MONTE we will describe is the implementation of the *cell list* [1] method in the program, with the aim of improving the general performance of the program for large systems.

2.1 Improving the scaling of neighbour list set-up

In molecular simulation, for short-range interactions pairs of atoms only interact if they are within a certain distance r_C, known as the *cut-off*, of each other. Thus it is standard practice to employ *neighbour lists* [1] to keep track of which atoms are within r_C of each atom j. The advantage of this is that when calculating, say, the energy of j, one already knows which atoms to consider in the calculation. The alternative is a costly search performed every energy calculation over the whole system for atoms within r_C of j. For a system comprised of N atoms, neighbour lists reduce the computational complexity associated with performing such an energy calculation from $O(N)$ to $O(1)$; the use of neighbour lists (or another scheme for avoiding system-wide searches) is hence essential for simulating systems with large N.

However, there is a cost associated with setting up neighbour lists, an operation done frequently during a simulation (e.g. at the outset of the simulation, and throughout the simulation to account for atoms' shifting positions). The simplest approach for determining the neighbour list for a given atom j involves a bruteforce search over the N atoms to determine those which are within a certain distance $R_{nl}=(r_C+r_V)$ of j, where r_V , known as a *Verlet shell*, is a small distance added to r_C to account for the fact that atoms could move to withinr_C of j between neighbour list updates, and hence it is insufficient to consider only atoms within r_C of j in constructing the list. Thus setting up neighbour lists for *all* atoms is an $O(N^2)$ operation.

For DL_MONTE the $O(N^2)$ scaling associated with setting up the neighbour lists becomes the bottleneck to simulating systems with very large N. *Cell lists* [1] provide a means for eliminating this bottleneck. In the cell list scheme the system, which we assume here for simplicity is cubic, is divided into smaller cubic cells with dimension $R_{cell,}$, and each cell is assigned atoms based on whether or not the positions of the atoms reside within the cell. If R_{cell} is chosen to be greater than R_{nl}, then for an atom j one can limit the search for atoms to add to j's neighbour list to: the cell to which j belongs; and cells adjoining the cell in which j belongs. By localising the search in this way, the complexity of setting up neighbour lists for all atoms reduces from $O(N^2)$ to $O(N)$.

We have implemented this scheme in DL_MONTE. The improvement to the scaling can be seen in Figures 1a) and 1b), which show the wall-clock time versus N for benchmark simulations designed to capture the performance of the neighbour list set up. These simulations consisted of initialisation, including initialisation of the neighbour lists, followed by two calculations of the total energy of the system using the neighbour lists. a) corresponds to the Lennard-Jones (LJ) [1] solid, and b) to fluorite ceria using the force field given in [9].(Input files for the simulations referred to in this report, and further technical details regarding these simulations, can be found at [5]). It is clear from these figures that the $O(N^2)$ scaling borne out when cell lists are not utilised – the aforementioned bottleneck for simulating systems with large N with DL_MONTE – is greatly improved upon by the use of cell lists, resulting in significant speed-ups in the benchmark simulations for large systems. For instance, the speed-up is a factor of ≈30 for the largest system we considered in the LJ solid. The more lacklustre speed-up of ≈2.3 for the largest ceria system reflects the fact that at such system sizes the Coulomb energy calculation, which scales as $\approx O(N^{1.5})$ and is unaffected by the use of cell lists, dominates.

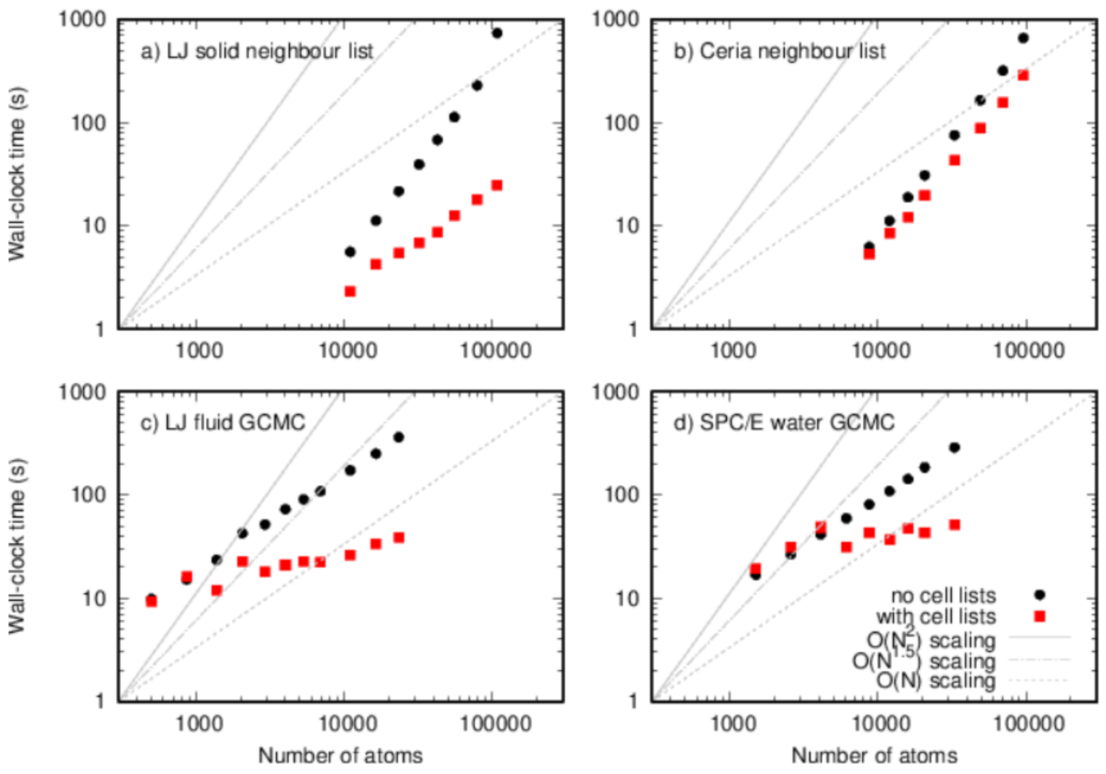

Figure 1: Scaling of various benchmark simulations versus system size (number of atoms, N), with and without the use of cell lists. The curves show the gradients which correspond to O(N), O($N^{3/2}$) and O(N^2) scaling with N.

2.2 Improving the scaling of GCMC simulations

As well as providing a means to improve the scaling associated with using neighbour lists, cell lists can also be used in their own right – without the use of neighbour lists – to locate atoms within j's interaction range during energy calculations. This entails choosing the dimensions of the cells to be $R_{cell} > r_C$, which ensures that during a search for atoms which interact with j during an energy calculation one need search only over the cells adjacent to, and including, the cell in which j is located. This approach is superior when neighbour lists would have to be set up very frequently during a simulation. One important situation where this is the case is GCMC, where neighbour lists would have to be set up every time an atom is inserted or deleted.

We have implemented this scheme in DL_MONTE. The improvement to the scaling can be seen in Figures 1c) and 1d), which shows the wall-clock time versus N for benchmark simulations designed to measure the performance of GCMC. Figures 1c) and 1d) correspond, respectively, to GCMC simulations for the LJ fluid, and liquid water modelled by the SPC/E force field [10]. For both, the wall-clock times reflect the time taken to perform a large number of single-molecule MC moves: molecule translations, rotations, insertions and deletions. It is clear from these figures that O(N) scaling is borne out when cell lists are not utilised, and that the scaling is greatly improved by the use of cell lists, yielding a significant speed-up for large *N.* For instance, for the largest systems we considered, the speed-up was a factor of ≈9 for the LJ fluid and ≈6 for SPC/E water.

3 Continuous fractional component Monte Carlo

3.1 Overview of method

In GCMC, insertion moves typically entail first inserting a new molecule into the system at a *random* position. The energy of the inserted molecule, *E*, is then calculated. Finally, the insertion is accepted or rejected with a probability which depends on *E* and certain prescribed thermodynamic properties of the system, namely the temperature, *T,* and chemical potential, μ, of the inserted molecule. Crucially, insertions are overwhelmingly rejected if *E* is very high, and the system evolves extremely slowly. This is troublesome for dense systems since randomly-chosen insertion positions are typically 'on top of' molecules already in the system, leading to high *E*, and hence low rates of successful insertions. In other words, for dense systems insertions are almost always rejected on account of the lack of 'cavities' in the system into which new molecules could be inserted with low *E*. For this reason GCMC and similar methods become intractable for systems in which the typical cavity size is much smaller than the size of the molecules to be inserted.

The aforementioned problem has been long recognised, and a number of methods, which we refer to here collectively as *continuous fractional component MC* (CFCMC) (see [11] for a recent review of the method), have been developed to address it. Consider first the problem of inserting *only one* new molecule *j* into the system. In CFCMC a quantity λ is associated with *j* which specifies the strength of the interactions between *j* and the rest of the system. λ=0 corresponds to interactions between *j* and the rest of the system being 'switched off': *j* is in effect absent from the system. On the other hand λ=1 corresponds to *j* fully interacting with the rest of the system: *j* is included in the system. All other values of λ, i.e. 0< λ<1, correspond to *j partially interacting* with the rest of the system in some manner which depends on the value of λ – something we elaborate on later. λ thus defines a *pathway* which, upon moving from λ=0 to λ=1, in effect results in molecule *j* being inserted into to the system. CFCMC involves treating λ as an additional degree of freedom in the system, and sampling *equilibrium configurations* for various λ in the range λ=0–1. To elaborate, at any λ, since we are sampling thermodynamic equilibrium, the positions of the other molecules in the system will be *relaxed* around molecule *j* in accordance with the inter-molecular interactions between *j* and the rest of the system corresponding to that particular λ. Thus as λ is increased from 0 to 1, and *j* is *gradually* inserted into the system, the system relaxes to accommodate the new molecule. In other words, by gradually inserting *j* in this manner a cavity is created 'naturally' which grows with *j* until *j* is inserted into the system fully at λ=1. Thus the problem of 'finding' a cavity encountered in conventional methods is bypassed.

We have implemented CFCMC in DL_MONTE, and in the rest of this section we describe this functionality in more detail, and conclude with an example application of the methodology.

3.2 Defining the pathway

One has considerable freedom choosing how *j* interacts with the rest of the system for $0< \lambda<1$, i.e. the choice of pathway. However, as one might anticipate, the choice of pathway has a bearing on the ease with which *j* can be inserted into the system. With this in mind, there are standard approaches employed in the literature, approaches which DL_MONTE's implementation of CFCMC accommodates. We now elaborate on this.

With regards to short-range (van der Waals) interactions, in molecular simulation the most commonly-employed form of interaction potential between atoms is the LJ potential [1]. Accordingly there is a λ-dependent generalisation of this potential which has found standard use in CFCMC [11]:

$$\phi(r,\lambda) = \lambda^a 4\epsilon\left[\left(\frac{1}{\alpha(1-\lambda)^b+(r/\sigma)^c}\right)^{12/c} - \left(\frac{1}{\alpha(1-\lambda)^b+(r/\sigma)^c}\right)^{6/c}\right]$$

This potential becomes the LJ potential at $\lambda=1$, at which point ε and σ have their usual significance. For $\lambda<1$ the potential softens as λ tends to 0, with the parameters *a, b, c* and α controlling the exact nature of the softening. At $\lambda=0$ the potential is simply 0. DL_MONTE supports this potential form for CFCMC, including tail corrections [1].

With regards to Coulomb interactions, DL_MONTE supports λ-dependent charges for *j*: the charge of atom *k* in molecule *j* can be specified to be $q_k(\lambda)=\lambda Q_k$, where Q_k is the 'true' charge of *k* when *j* is fully interacting with the rest of the system. By allowing the charges in the molecule to thus scale linearly with λ, the Coulomb interactions with the rest of the system can be gradually turned on as λ is varied from 0 to 1.

For interatomic potentials comprised of both short-range and Coulomb interactions it is standard practice to switch on the short-range interactions before the Coulomb interactions as λ is increased [11]. To accommodate this, we first define a value of λ at which the *handover* from the short-range interactions to the Coulomb interactions occurs, $\lambda_{handover}$. We then define separate λ quantities pertaining to each type of interaction; we denote the λ pertaining to the short-range and Coulomb interactions as λ_{sr} and λ_C, respectively. Their values depend on the 'global' λ as follows:

- $\lambda<\lambda_{handover}$ corresponds to the range of λ where the short-range interactions are switched on: $\lambda_{sr}=0$ at $\lambda=0$, increasing to $\lambda_{sr}=1$ at $\lambda=\lambda_{handover}$. Moreover, $\lambda_{sr}=1$ for $\lambda>\lambda_{handover}$.
- $\lambda>\lambda_{handover}$ corresponds to the range of λ where the Coulomb interactions are switched on: $\lambda_C=0$ at $\lambda=\lambda_{handover}$, increasing to $\lambda_C=1$ at $\lambda=1$. Moreover, $\lambda_{sr}=0$ for $\lambda<\lambda_{handover}$.

By tuning $\lambda_{handover}$, one can improve the efficiency of the pathway.

3.3 Sampling the pathway

In order to sample configurations with different λ, MC moves which vary λ must be employed alongside conventional MC moves (e.g. atom translation, molecule translation and rotation, volume expansion/contraction). One has considerable freedom with regards to the form of such 'λ moves', and different problems require different types of λ move [11]. We have implemented two approaches in DL_MONTE, which we will now describe.

Basic λ move

DL_MONTE supports *discrete* values of λ; λ is allowed to take one of n possible *states* between and including λ=0 and λ=1. Specifically, the ith discrete state is $\lambda_i=(i-1)\Delta\lambda$, where $\Delta\lambda=1/(n-1)$ is the separation in λ between states. Note that the first and *nth* states are the *physical* states $\lambda_1=0$ and $\lambda_n=1$. With this in mind, the basic λ move which DL_MONTE supports attempts to change λ by $\mp\Delta\lambda$, where moves which take λ below 0 or above 1 are rejected.

Calculating chemical potentials

One application of this type of move is to calculate chemical potentials. To elaborate, for a system at prescribed temperature T, number of molecules N, and volume V, the chemical potential, μ, is defined as the free energy change of the system upon adding an additional molecule to the system. This is related to the free energy difference between the λ=1 and λ=0 states in the above pathway via

$$\mu = \mu_{IG} + \mu_{ex}$$

and

$$\mu_{ex} \equiv \left[F(\lambda=1) - F(\lambda=0)\right]$$

where μ_{IG} is the ideal gas chemical potential for the system – which is a known analytical function of N, V and T – and μ_{ex} is known as the *excess chemical potential.* μ_{ex} can be calculated by using the aforementioned basic λ moves to sample all λ states, and comparing how often the λ=0 and λ=1 are sampled. In this manner, the basic λ moves enable μ to be calculated.

Free energy methods

Often the pathway parametrised by λ exhibits free energy barriers which make movement between λ=0 and λ=1 very slow, in turn making an accurate determination of μ_{ex} intractable. In this case *free energy methods* must be used to accelerate sampling over λ to enable μ_{ex} to be calculated; see [4] for further details regarding free energy methods with regards to DL_MONTE. DL_MONTE supports a wide range of free energy methods [4] (e.g. Wang-Landau, transition-matrix Monte Carlo) and we have extended DL_MONTE's free energy functionality so that these methods can also be applied to sampling over λ. We demonstrate the use of free energy methods in conjunction with basic λ moves to calculate a μ_{ex} in a moment.

Insertion/deletion λ moves

The other type of λ move supported in DL_MONTE, which we refer to as an *insertion/deletion λ move*, is similar to the basic λ move describe above, except that its behaviour is different if the move takes takes λ above1 or below 0. Let us assume that the system is currently comprised of N 'real' molecules and 1 fractional molecule, j. In an insertion/deletion λ move, if the move takes λ above 1 then the fractional molecule j is 'promoted' to become a real molecule, and a new fractional molecule with λ=0 is inserted into the system at a random position. Thus the move, if accepted, yields a configuration with λ=0, and (N+1) real molecules and 1 fractional molecule in the system; an extra molecule has been inserted. Consider now what happens if an insertion/deletion λ move takes λ below 0. In this case the fractional molecule j is deleted, and a randomly-chosen real molecule is 'demoted' to become a fractional molecule with λ=1. Thus the move, if accepted, yields a configuration with λ=1, with (N-1) real molecules and 1 fractional molecule in the system; a molecule has been deleted.

Insertion/deletion λ moves enable the benefits of gradual insertion over the λ pathway to be exploited to insert *multiple* molecules into the system analogously to GCMC. In fact, like GCMC, insertion/deletion moves are accepted with a probability which reflects the energy change of the system upon insertion/deletion, as well as the prescribed T and μ.

3.4 Example application: SPC/E water

To demonstrate and validate DL_MONTE's CFCMC functionality, we considered the SPC/E water model in the liquid phase at 300K and the saturation pressure, in a simulation box of volume 8000Å^3. Note that this model contains both short-range (including tail corrections) and Coulomb interactions, making it an ideal test of the CFCMC functionality. First we calculated μ_{ex} at these conditions using basic λ moves at fixed N, V and T, in conjunction with free energy methods, as described above. Moreover, we considered two possible λ pathways: $\lambda_{handover}$=0.5 and $\lambda_{handover}$=0.7. The free energy versus λ obtained from these calculations is shown in Figure 2 (red and blue points). Note that, as expected, $[F(\lambda=1)-F(\lambda=0)]=\mu_{ex}$ does not depend on the choice of pathway. Moreover, μ_{ex} is in agreement with the literature value [12] (dotted line).

We also simulated this system at fixed μ, V and T using λ insertion/deletion moves, again in conjunction with free energy methods, to sample different numbers of molecules, N, in the system. A trajectory of such a CFCMC simulation is shown in Figure 2 (inset, green curve). Here, (N+λ) is plotted versus number of MC moves. Also shown for comparison is N versus number of MC moves for a conventional GCMC simulation for the same system. The CFCMC simulation yields $<N>$=266.4(6), which is in agreement with the GCMC simulation [267.4(5)] and the literature value [266.9(8)] [12]. Moreover, the free energy profile 'learned' during the simulation to apply to λ moves is in agreement with the NVT results, as can be seen in the figure (black points).

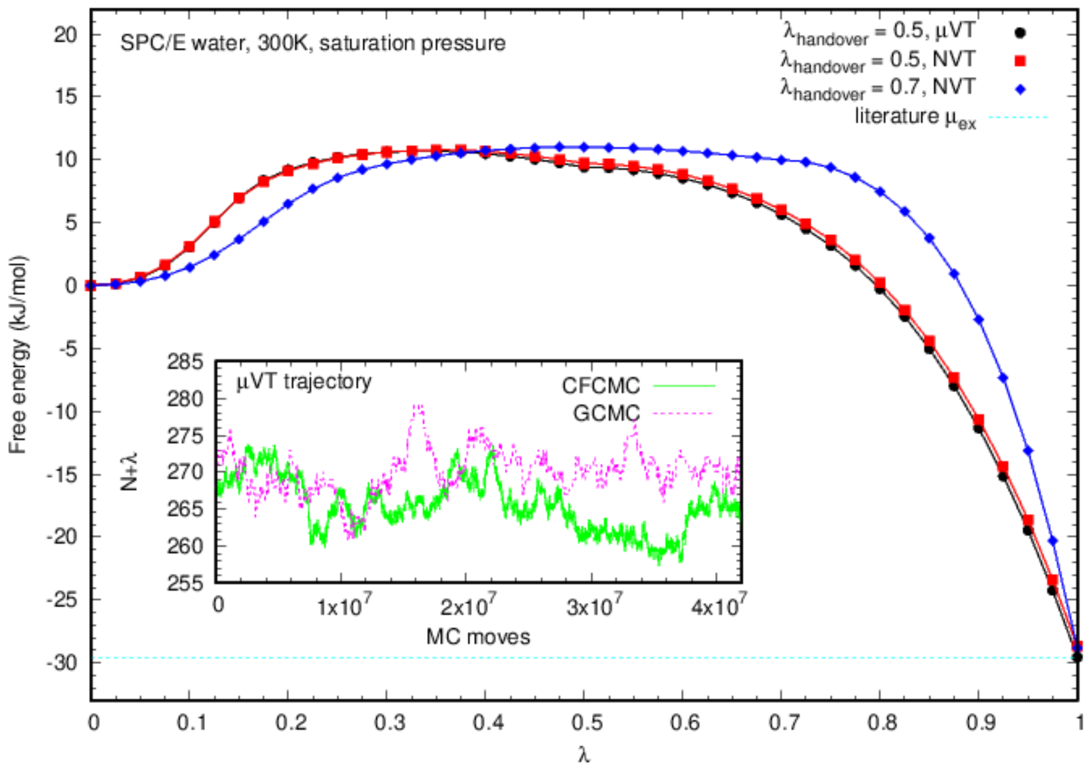

Figure 2: Results from CFCMC simulations of SPC/E water in the liquid phase at 300K and the saturation pressure. The main plot shows the free energy versus λ obtained from various CFCMC simulations, along with the literature μ_{ex} from [12]. The inset shows (N+λ) versus number of MC moves for a CFCMC simulation, and N versus number of MC moves for a GCMC simulation, at the aforementioned conditions, where N is the number of water molecules in the system.

4 Parallelisation performance

MPI parallelisation can be employed in DL_MONTE to realise a speed-up by reducing the wall-clock time associated with performing energy calculations. Moreover, DL_MONTE also employs MPI parallelisation during simulations utilising the replica exchange Monte Carlo (REMC) method. REMC [2] entails simulating multiple replicas of the system at different, say, temperatures, simultaneously, while occasionally employing a *replica exchange MC move* to exchange configurations between pairs of replicas at nearby temperatures. The benefit of this approach is that it results in low-temperature replicas exploring configurations which would take a very long time to sample using conventional methods.

We have made general improvements to the MPI parallelisation of DL_MONTE, both with regards to energy calculations and replica exchange. These improvements included reorganising parallelised loops involved in energy calculations, extending parallelisation to the new cell list functionality, and removing bottlenecks inhibiting the parallel efficiency of the program. In this section we provide an overview of the parallelisation set-up in DL_MONTE, and quantify the performance of DL_MONTE on ARCHER 2 following our improvements.

4.1 Overview of parallelisation set-up

Key components of the energy in DL_MONTE are the reciprocal contribution to the Coulomb energy, E_{re}; and what we will refer to here as the *two-body energy*, E_{2b}, which is comprised of the short-range 'van der Waals' energy, the real-space component of the Coulomb energy, and the energy associated with intra-molecular bonds. The parallelisation of energy calculations in DL_MONTE involves employing multiple MPI processes to speed-up the tasks of evaluating E_{re} and E_{tb}. Calculating E_{rec} entails summing over energy contributions from a large number of reciprocal lattice vectors. To parallelise this, DL_MONTE partitions the reciprocal lattice vectors into subsets, and assigns the component of the calculation associated with each subset to a different process. After each component has been calculated, E_{rec} is obtained by summing the contributions from each process via an MPI_ALLREDUCE.

The parallelisation of the two-body energy calculation is more complex. Calculating E_{2b} entails summing over energy contributions associated with many pairs of atoms. To parallelise a calculation of E_{2b}, DL_MONTE will partition the set of atom pairs into subsets, and different MPI processes will calculate the energy contribution from each subset. MPI_ALLREDUCE is then used to sum the contributions from all subsets to obtain E_{2b}. The scheme used to create the subsets depends on numerous factors, in particular:

- the size of the component of the system whose energy is to be calculated, e.g. atom, molecule, or the whole system;
- whether or not cell or neighbour lists are in use;
- whether or not a directive '*paratom'* is included in the CONTROL file: this directive changes the manner in which atoms involved in the calculation are assigned to different processes.

With regards to REMC, in the simplest instance DL_MONTE would employ N_{rep} MPI processes to simulate the N_{rep} replicas under consideration in parallel – one process per replica. In this case, the simulation would entail all replicas periodically performing replica exchange MC moves simultaneously. During these moves, there is communication between processes to decide which pairs of replicas should exchange configurations. Moreover, there is transfer of configuration data between processes involved in configuration exchanges. These MC moves are performed infrequently; most of the simulation entails *independent* evolution of the replicas by their respective processes via conventional MC moves (e.g. atom/molecule translation, volume moves).

It is also possible to employ REMC in conjunction with parallelisation of energy calculations. In this case, p_{eng} processes are assigned to each of the N_{rep} replicas for the purposes of speeding up their energy calculations, bringing the total number of processes employed in the REMC simulation to $N_{rep}p_{eng}$.

4.2 Performance and MPI scaling

Parallelisation of energy calculations

The speed-up of various benchmark simulations versus number of MPI processes p_{eng} is shown in Figure 3. These simulations were all performed on ARCHER 2. As mentioned earlier in this work, an MC simulation consists of evolving the system by repeatedly applying various types of MC move. Most move types involve changing the degrees of freedom associated with a single atom or molecule: *atom/molecule moves*. By contrast, *volume moves* involve changing the entire system. To add insight into the scaling of volume moves, two of the benchmark simulations use *only* volume moves: these benchmarks are labeled '*vol. moves*'. The remaining benchmarks use only atom/molecule moves. It can be seen from the figure that the simulations which use only volume moves scale better with number of MPI processes than those which use atom/molecule moves. This highlights the fact that it is atom/molecule moves which are the bottleneck to the scalability of DL_MONTE: the maximum possible speed-up we observed using atom/molecule moves was a factor of ≈6 for the water GCMC benchmark simulation.

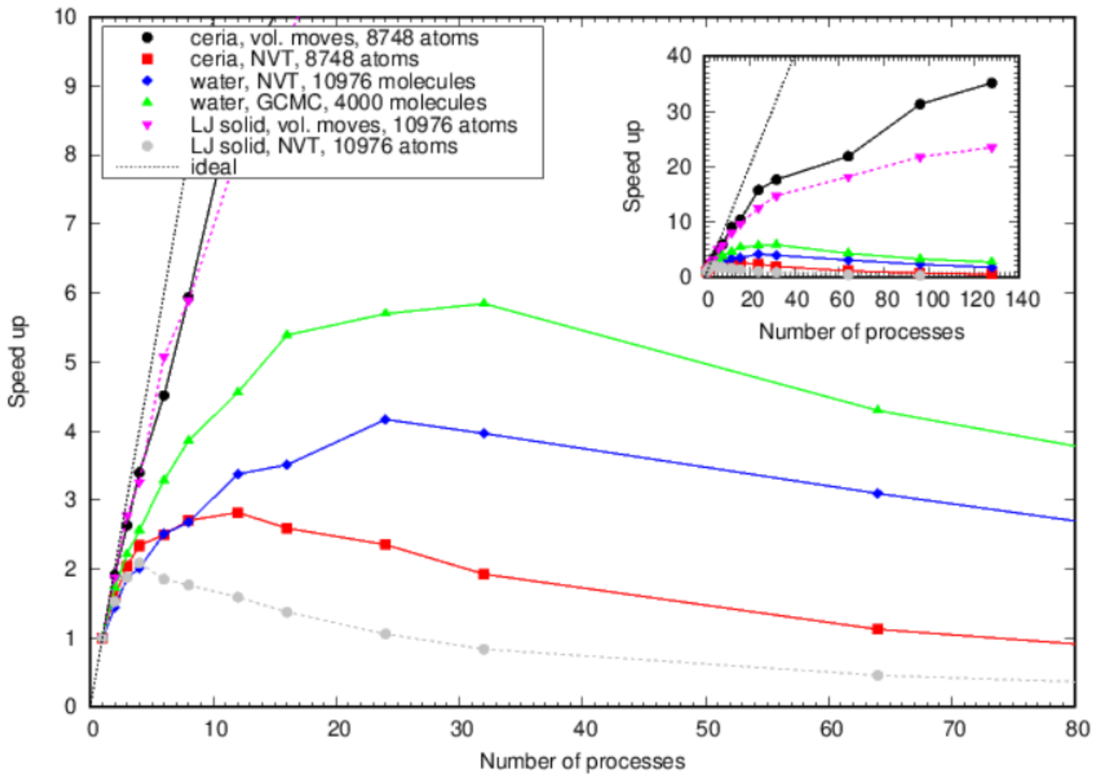

Figure 3: Speed-up of various benchmark DL_MONTE simulations versus number of MPI processes.

The reason for this is that atom/molecule moves have a far smaller computational load than volume moves with regards to their energy calculations, and hence they more quickly reach the point where the time a process spends calculating its contribution to the energy becomes comparable to the time spent communicating between processes – at which point using further processes is counterproductive. Moreover, because the computational load associated with the energy calculation is smaller than that of the volume move, the *serial* overhead associated with the atom/molecule move (e.g. generating the new position of the molecule, deciding if the move should be accepted) is a greater fraction of the CPU time than for the volume move. This, according to Amdahl's law, means that the highest possible speed-up for atom/molecule moves will be lower than for volume moves. On the other hand, it is clear from the above discussion that the scalability would improve if the computational load associated with the energy calculation in atom/molecule moves were to increase. This would be the case if, for instance, the interatomic potentials became more costly to evaluate, or, for the case of molecule moves, the number of atoms in the molecule were to increase. This is borne out in Figure 3: as well as having a more costly force field, the SPC/E water has more atoms per molecule than the LJ solid, and hence yields a higher maximal speed-up. Thus we would expect DL_MONTE to scale better for more computationally expensive force fields.

Replica exchange Monte Carlo

Figure 3 suggests that using more than ≈30 MPI processes is unlikely to yield further speed-up in typical DL_MONTE simulations. However, the prospect still exists to exploit large numbers of processes to simulate many replicas of the system simultaneously via REMC. Figure 4 shows the wall-clock times for benchmark REMC simulations versus N_{rep}. The simulations were performed on ARCHER 2, and all utilised the same number of MC moves per replica. Moreover, various degrees of parallelisation of the energy calculations within the replicas were considered: p_{eng}=1 (no parallelisation), 2, 4 and 16. Note that the total number of processes, $p_{eng}\,N_{rep}$, utilised by our largest simulation is 10,240, which corresponds to 80 nodes on ARCHER 2.

With ideal scaling the wall-clock time would not increase with N_{rep} for a given p_{eng}. Clearly ideal scaling is far from realised. However, the loss in CPU time associated with this inefficiency may be offest by the improvement in sampling which comes from employing more replicas in the REMC method. Interestingly the speed-up in the atom/molecule moves which comes with using, say, p_{eng}=16 as opposed to p_{eng}=1 holds even at very high numbers of replicas. It is therefore viable to employ REMC up to N_{rep}~1000, even in conjunction with parallelisation of the energy calculations.

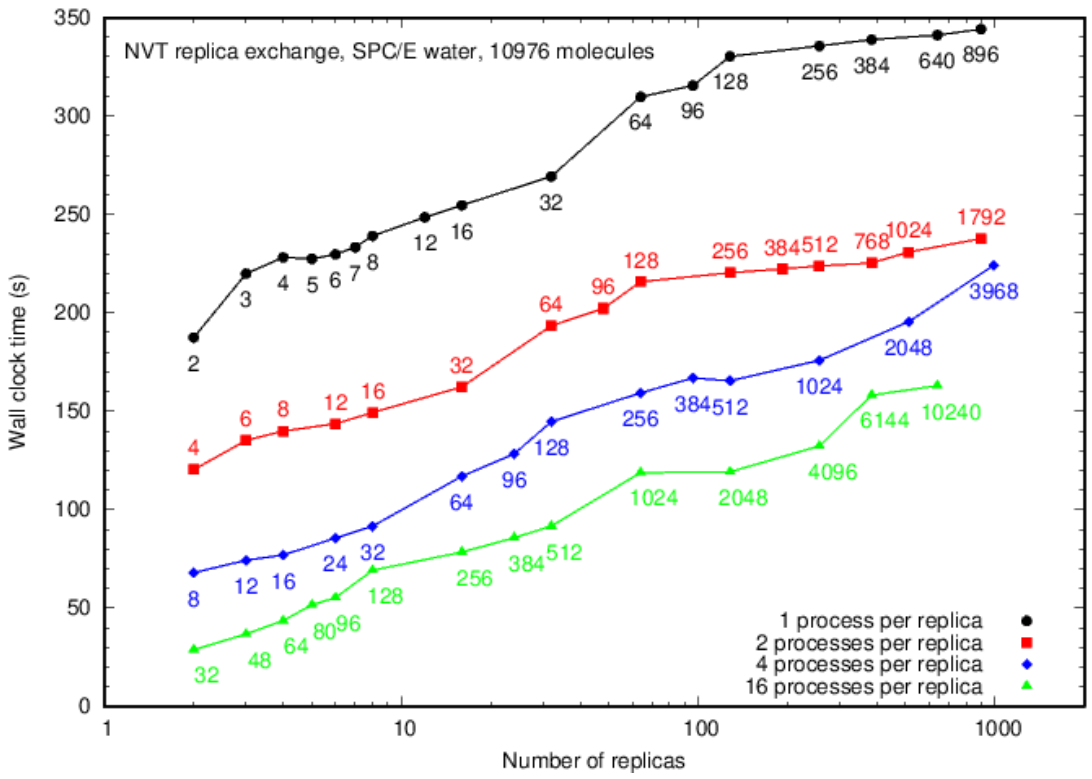

Figure 4: Scaling of DL_MONTE NVT replica exchange Monte Carlo simulations versus number of replicas, for different numbers of processes used to parallelise the energy calculations within each replica. Each data point is labeled with the total number processes the simulation utilised, i.e. p_{eng} N_{rep}.

5 Summary

We have made various improvements to the general-purpose Monte Carlo simulation program DL_MONTE to facilitate its application to *large-scale* simulations, i.e. simulations of systems comprised of large numbers of atoms or complex molecules; or simulations which exploit a large number of parallel processes. The improvements we made are as follows.

Firstly, we implemented the *cell list* method in DL_MONTE. This method entails dividing the system into cells, and keeping track of which cell each atom is in at all times, as a means to speed up searches for pairs of atoms which are within a certain distance of each other. We demonstrated that using cell lists in DL_MONTE yielded significant speed-ups in large systems.

Secondly, we implemented the *continuous fractional component Monte Carlo* method in DL_MONTE. This method enables insertion of molecules into the system during a simulation – something routinely employed in Monte Carlo simulation studies of adsorption and fluid properties – in situations where standard approaches are intractable, e.g. solutions in which the solvent and solute molecules have a large size difference. We demonstrated and validated this new functionality by applying it to a commonly-used model of water.

Finally, we made general improvements to the performance and scalability of the MPI parallelisation in DL_MONTE, both the parallelisation invoked to speed up the energy calculations, and the parallelisation utilised by DL_MONTE to perform replica exchange Monte Carlo (REMC) simulations – a technique in which many copies of the system are simulated in parallel. After making these improvements, we quantified the parallel performance of DL_MONTE on ARCHER 2 for a range of simulation types. One notable outcome of this was that it is viable to invoke DL_MONTE with ≈10,000 processes to simulate ≈1000 copies of the system via REMC. This paves the way to using such large numbers of processes to solve real-world problems with REMC using DL_MONTE.

The improvements to DL_MONTE described here will be included in the forthcoming release of DL_MONTE, v2.08. Further details on how to obtain the program, as well as other documentation, can be found at [5].

References

[1] B. Smit & D. Frenkel, 'Understanding Molecular Simulation: From Algorithms to Applications', London: Academic Press (2002)
[2] R. H. Swendsen & J.-S. Wang, Phys. Rev. Lett. 57, 2607 (1986)
[3] J. A. Purton, J. C. Crabtree & Parker S C, Mol. Sim. 39, 1240 (2013)
[4] A. V. Brukhno et al., Mol. Sim. 47, 131 (2021)
[5] https://gitlab.com/dl_monte/
[6] T. L. Underwood et al., arXiv:2104.03822 [physics.comp-ph]
[7] R. J. Gowers, A. H. Farmahini, D. Friedrich & L. Sarkisov, Mol. Sim. 44, 309 (2018)
[8] Y. Nejahi et al., Software X 9, 20 (2019)
[9] T. X. T. Sayle et al. Nanoscale 5, 6063 (2013)
[10] H. J. C. Berendsen, J. R. Grigera & T. P. Straatsma, J. Phys. Chem. 91, 6269 (1987)
[11] A. Rahbari et al., Mol. Sim. 47, 804 (2021)
[12] https://www.nist.gov/mml/csd/informatics/sat-tmmc-liquid-vapor-coexistence-properties-spce-water-lrc, accessed August 2022

Acknowledgements

This work was funded under the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (http://www.archer2.ac.uk). This work also made use of the Balena High Performance Computing Service at the University of Bath.