Journal of Computational and Applied Mathematics 433 (2023) 115278

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Towards auto-tuning Multi-Site Molecular Dynamics simulations with AutoPas

Samuel James Newcome [a,*], Fabio Alexander Gratl [a], Philipp Neumann [b], Hans-Joachim Bungartz [a]

[a] School for Computation, Information and Technology, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany
[b] Fakultät für Maschinenbau, Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany

ARTICLE INFO

Article history:
Received 1 November 2022
Received in revised form 27 March 2023

Keywords:
Molecular Dynamics
High performance computing
Auto-tuning
AutoPas
Multi-Site Molecular Dynamics

ABSTRACT

There exists an extensive literature of algorithms for short-range pairwise interactions in particle simulations, however, there is no single algorithm that performs the most optimally in every scenario, motivating the use of auto-tuning to select the optimal pairwise interaction algorithm. Previous efforts to auto-tune Molecular Dynamics have focused on Single-Site Molecular Dynamics, where the computational cost for the intermolecular force calculation is constant. Alternatively, for Multi-Site Molecular Dynamics, the cost of this calculation varies with the number of sites, which, as we show in this paper, can result in different optimal algorithms. Despite this further benefit for auto-tuning, it has yet to be applied to Multi-Site Molecules. In this paper, we introduce an implementation of Multi-Site Molecular Dynamics that is integrated with AutoPas. Using this implementation, we analyse how the relative performance between these algorithms varies as the number of sites varies, for both homogeneous and heterogeneous molecule distributions, and for two different hardware. Furthermore, we demonstrate the advantage of auto-tuning in the context of Multi-Site Molecular Dynamics using the node-level short-range particle simulation library, AutoPas.

1. Introduction

Molecular Dynamics (MD) simulations are a useful tool in a variety of scientific areas, ranging from thermodynamics [1] to protein folding [2], where continuum models are not powerful enough. Rather than simulating macroscopic quantities, such as densities and temperatures, in MD one aims to simulate the molecules themselves and their interactions — from which macroscopic quantities can be calculated.

MD simulations are a classical example of an N-body problem, with each molecule represented by a body of point sites. In a coarse-grained molecular model, this body might simply be a single point site, however, for greater accuracy of simulation, multi-site simulations might be required. The intermolecular dynamics of a simulation are typically determined using the sum of forces acting between intermolecular pairs of sites. For N molecules with $O(s)$ sites each, this results in $O(s^2N^2)$ force calculations. For simulations with large numbers of molecules, for example in thermodynamics applications [3], this computational complexity is not affordable.

For short-range forces, such as the one arising from the Lennard-Jones potential, a common method to reduce simulation time is to introduce a cutoff r_c, such that the force contributions for pairs of molecules with centre-of-masses

* Corresponding author.
E-mail address: samuel.newcome@tum.de (S.J. Newcome).

https://doi.org/10.1016/j.cam.2023.115278
0377-0427/

further apart than r_c is 0. This reduces the number of force calculations to $O(s^2N)$ for many relevant scenarios, however, the number of distance calculations is still $O(N^2)$.

To reduce this to $O(N)$, *Particle Containers*, which store some additional coarse spatial information, are used. Examples of these include *Linked Cells* [4] and *Verlet Lists* [5] — in the Linked Cells algorithm, molecules are sorted into spatial cells and interacted only with those in neighbouring cells and, in the Verlet Lists algorithm, lists of neighbouring molecules are created, and molecules are only interacted with those in their neighbour list.

In this paper, we will compare the performance of these algorithmic choices, as well as others, for multi-site molecular models with different numbers of sites, for different molecular distributions, and for different hardware, demonstrating that no algorithmic choice is superior to all others in every scenario. In particular, we analyse how the optimal algorithmic choice varies as the number of sites varies.

The idea that there is no optimal algorithm in every scenario motivated the development of the node-level auto-tuning particle simulation library, AutoPas [6].[1] The aim of AutoPas is to abstract the algorithmic choice away from the user, as well as dynamically select the optimal algorithm over the course of the simulation. Furthermore, it can be used to select different optimal algorithms in different regions of space [7].

We have recently provided support for multi-site molecular models using AutoPas, massively increasing the applicability range of the software and its auto-tuning methodology, for example in process engineering [8]. In this paper, we will demonstrate the benefits of using AutoPas to select the optimal algorithm for multi-site MD simulations.

Whilst multi-site molecular models are not novel, to the authors' knowledge these have never before been used with automated algorithm selection methods, and there is no other known software which provides such black box auto-tuning support for this type of MD simulation. As such methods are expected to address the problem of hardware-independent and simulation-independent optimal performance portability, the use of such methods holds significant importance for this field in the era of exascale computing.

2. Algorithms for short-range particle simulations

As a result of the significant time cost for the pairwise force calculation in MD simulations, many methods have been developed to efficiently evaluate this. This paper will focus on two categories of methods: *Particle Containers* and *Traversals*. Particle Containers dictate how, and with what information, the particles are stored, primarily to efficiently reduce the number of distance calculations that lead to force contributions. Traversals detail the manner in which the particle containers are traversed, which is of particular importance when applying shared-memory parallelism so that multiple threads can process the pairwise interactions within the container at the same time, whilst guaranteeing no race conditions.

2.1. Particle containers

Particle Containers store the particles either in some coarse spatial structure and/or with other information about their environment. The design of such a method has to balance the advantages of reducing redundant distance calculations and good memory alignment with the disadvantages of computational and memory overhead. Discussed in this paper are *Linked Cells* which store the particles in spatial cells, *Verlet Lists* which store the particles with lists of other particles in their neighbourhood, and *Pairwise Verlet Lists* which combine both containers.

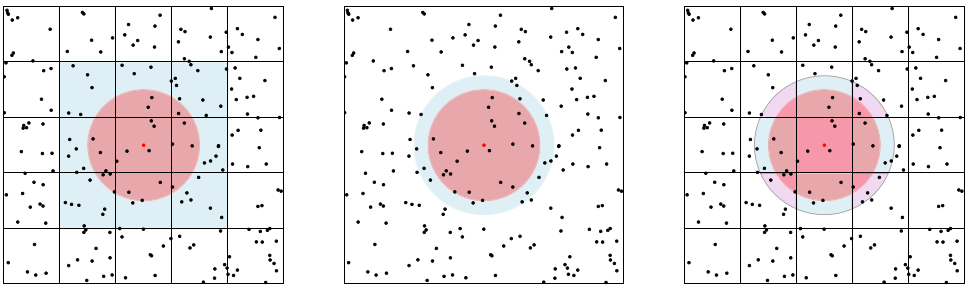
2.1.1. Linked cells

The Linked Cells (LC) algorithm divides the domain spatially into cells. Molecules in a cell interact with those inside their own cell, and with those in neighbouring cells, as depicted in Fig. 1(a). The advantage of Linked Cells is their small memory and computational overhead, as the only overhead is maintaining the cell structure.

It is due to this low memory requirement that world records for the largest MD simulations have been broken using *ls1-mardyn* with its native Linked Cells algorithm [9,10]. A further advantage of the sorting of molecules is that vectorisation is relatively simple, as interacting two cells consists of interacting molecules consecutive in storage.

The significant disadvantage of Linked Cells is the amount of redundant distance calculations. Consider a homogeneous molecule distribution divided into cells with all dimensions of length r_c. A molecule only needs to interact with those in its own cell and its immediate neighbours — a total volume of $27r_c^3$. Yet the volume of the cutoff sphere is $\frac{4}{3}\pi r_c^3$. Thus, only $\frac{4}{81}\pi \approx 15.5\%$ of distance calculations result in a force contribution. If we increase the box dimensions, this percentage becomes worse however we have to rebuild the data structure less frequently. If we decrease the box dimensions, increasing the number of neighbouring cells appropriately to still cover all the cutoff sphere, this percentage increases, but comes at the additional overhead of having to access more cells and a greater overhead of managing the data structure.

[1] https://github.com/AutoPas/AutoPas



(a) Linked Cells (b) Verlet Lists (c) Pairwise Verlet Lists

Fig. 1. Visualisation of Particle Containers implemented in AutoPas. The red circle denotes the cutoff sphere. The area in blue and pink denotes the region outside of the cutoff for which the Particle Container performs distance calculations. For Verlet Lists and Pairwise Verlet Lists, this is also the molecules stored in the neighbour list, with altering blue and pink representing different lists.

2.1.2. Verlet lists

The Verlet List (VL) algorithm begins by determining a neighbour list, for each molecule, of molecules within a larger sphere of radius $r_c + \Delta_s$, where Δ_s is called the *Verlet Skin*. In future timesteps, a molecule is only interacted with those in its neighbour list, and, for every N timesteps, the neighbour list is rebuilt. Δ_s and N should be chosen such that no molecule can pass from greater than $r_c + \Delta_s$ away, to within r_c away, without the neighbour list being rebuilt.

The advantage of this is, typically, far fewer redundant distance calculations. This is, of course, subjective to Δ_s, which is naturally smaller in simulations with smaller distances travelled per timestep — whether through being a slower simulation or smaller timesteps. Outside of this case, a smaller Δ_s can be used, if one accepts that the lists will have to be rebuilt more frequently.

The significant disadvantage of Verlet lists is the far higher memory consumption, due to each molecule having to store a list of every neighbouring molecule. Furthermore, having molecules stored in a random order leads to slower performance.

2.1.3. Pairwise verlet lists

Pairwise Verlet Lists (PVL), proposed by Gonnet [11], combine both previous particle containers. As can be seen in Fig. 1(c), molecules are sorted into spatial cells, like Linked Cells. However, each molecule also has a set of neighbour lists, one for its own cell and every cell interacted with. For example, with cell dimensions at least $r_c + \Delta_s$ long, this would be 27 neighbour lists for each molecule.

The advantage of this is the fewer redundant distance calculations of Verlet Lists, combined with the close proximity of data for molecules in the same neighbour list. Sorting the molecules into cells also allows the use of cell traversals, as discussed in Section 2.2. The disadvantage of this is the greater overhead coming from maintaining a cell structure, plus 27 different neighbour lists per molecule. This disadvantage causes particular problems in sparse simulations, where individual neighbour lists hold only one or two molecules.

2.2. Traversals

Shared-Memory parallelisation is a well-established technique for improving the efficiency of MD simulations [12]. Care must be taken to assign force calculations to threads efficiently without race conditions. For cells-based algorithms, in particular, a range of different *Cell Traversals* have been developed [6]. We will focus on three in particular: C01, C08, and C18.

In the **C01** traversal, each cell is processed independently. For all molecules in a cell, a single thread calculates all force calculations with other molecules in the cell and those in neighbouring cells. Pairwise force calculations contribute to forces and torques only for molecules within said cell. With each cell being processed independently, this traversal is highly parallelisable.

As per Newton's Third Law (N3L), we can do half the number of force calculations by taking advantage of the fact that the negation of the force of site A upon site B is the force of site B upon site A. This can, for C01, only be used for the force calculations within the same cell.

An alternative traversal, which can use N3L for all calculations, is the **C18** traversal. Here, cells are assigned one of 18 colours, in a repeating pattern (see Fig. 2(b)). For each colour, threads can independently calculate force contributions for pairs of molecules internal to cells of that colour, as well as force calculations occurring between that cell and cells with index one higher. Provided threads synchronise such that threads only work on cells of one colour, no race conditions occur. With 18 synchronisation points but larger chunks of work available, this has a worse parallelisability than C01 but a smaller scheduling overhead.

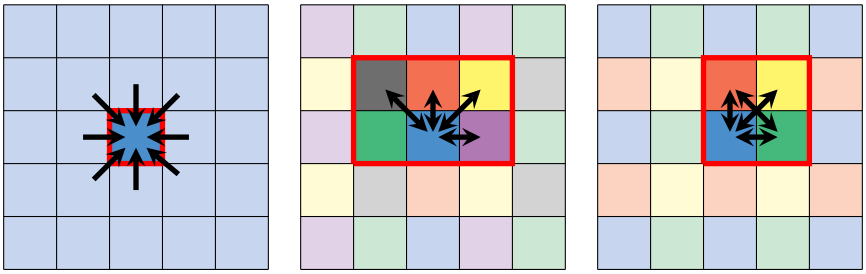


Fig. 2. Visualisation of the Cell Traversals C01, C18, C08. Threads can freely work with primary cells of one colour at a time, and their neighbours, as dictated by the arrows. Synchronisation points are implemented such that only one thread can update molecules within the red box. The single-headed arrows in the C01 figure represent the inability to use N3L optimisations.

The **C08** traversal takes the same approach but applies it for an 8-colour-only scheme. As illustrated in Fig. 2(c), the trick to doing this is a cell–cell interaction which does not include the base cell. With only 8 colours, and thus only 8 synchronisation points, this has better parallelisability than C18, albeit not as good as C01.

3. Related work

Aside from multi-site MD, short-range particle simulations have a large diverse range of applications across science and engineering. In Section 3.1, we present some of these applications, with a particular focus on those which could benefit from using different algorithms over the course of the simulation or in different regions of the simulation.

With such importance to science and engineering, there is a significant body of work comparing methods for this type of simulation. In Section 3.2, we discuss works comparing the algorithms presented in Section 2, as well as their limitations.

3.1. Applications of short-range particle simulations

Algorithms for short-range particle simulations have numerous applications, including Discrete Element Methods [13], Smoothed Particle Hydrodynamics [14], Space Debris simulations [15], and Molecular Dynamics [4]. Whilst the use cases of these simulation techniques vary dramatically, they all consist of integrating Newton's equations of motion and involve a computationally dominant interaction between particles.

In particular, in such applications, it is common to see situations where the distribution of particles varies over time or space. Zhao et al. discuss applications of the Discrete Element Method in agriculture, with examples including the ploughing of earth, where the distribution of soil particles changes after being ploughed [16]. In their review paper on the applications of Smoothed Particle Hydrodynamics, Monaghan describes interfaces between regions of high-density particles and low-density particles, where the higher-density particles move into the low-density region [14]. Eckelsbach and Vrabec study vapour–liquid equilibria with multi-site MD, by simulating binary mixtures of molecules with different densities and numbers of sites [8].

The use of AutoPas can aid various applications. Simulations that show significant changes over time can utilise the dynamic tuning feature of AutoPas. Additionally, simulations with varying particle distributions in space can benefit from using different algorithms specific to different regions of space.

3.2. Works on comparing algorithms for short-range particle simulation

Linked Cells and Verlet Lists have been compared extensively. Domínguez et al. have provided an extensive comparison between Linked Cells and Verlet Lists for Smoothed Particle Hydrodynamics, including a further extension of the Verlet List method presented above [17]. However, this comparison does not take into account the benefits of parallelism.

Tchipev et al. provide a comparison between the native Linked Cells C01, C08, and C18 implementations of ls1-mardyn [12]. Seckler et al. extend this by comparing the performance of a much wider range of methods using ls1-mardyn with the non-native AutoPas backend [7]. This paper focuses more on the implementation of AutoPas with ls1-mardyn, and the benefits of dynamically tuning within that simulator — and so the comparison provided is limited in scope. Furthermore, this paper only investigates single-site MD.

Finally, Gonnet compares Linked Cells, Verlet Lists, and Pairwise Verlet Lists, in their paper introducing the latter method, for both a single-site Argon model and a 3-site Water model [11]. This includes comparison of parallel efficiencies and speedup as the number of cores increases. It was demonstrated that for low particle-densities and low number of

cores, Verlet Lists perform best, for high densities and high numbers of cores Linked Cells performed best, and the Pairwise Verlet Lists performed well in-between these extremes.

Although extensive comparisons have been made among these algorithms, there are still inadequate areas in the comparison process, especially when exploring the performance of these algorithms with varying site numbers. Nevertheless, the analysis indicates that there is no single algorithm that performs optimally in all situations, which underscores the need for utilising AutoPas.

4. AutoPas

As demonstrated previously [6,7], the fastest algorithm may change over the course of a simulation and different regions of a simulation may benefit from different algorithmic choices.

Such a concept motivated the development of the particle simulation library, AutoPas, which aims to dynamically select the fastest algorithm over the course of the simulation [6]. Such algorithmic choices include the particle container and traversal, as discussed above, as well as further choices such as data layout or the size of cells for cells-based particle containers. In this paper, we shall only consider tuning over the algorithms presented in Section 2. For other algorithmic choices available in AutoPas, as well as greater details, the reader is referred to the release paper of AutoPas [6].

4.1. Design and use of AutoPas

AutoPas is a library, written in C++17, that is designed as a black box upon which other particle simulation software can be built. The user creates classes which implement the particle they wish to simulate and the pairwise interaction between particles, called *Functors*. Both user-created classes must implement an interface required by AutoPas, but can do so by inheriting from base classes provided within the library.

Within the library, an example implementation of a single-site molecule has been provided, as well as example implementations of the Lennard-Jones force as a functor. As discussed in Section 5, these examples have been extended to include example implementations of multi-site molecules and multi-site Lennard-Jones force calculations.

At the beginning of a simulation, the user's code creates an AutoPas container object, templated with the user's particle type and pairwise interaction classes. The user can then set relevant simulation parameters such as the cutoff, which algorithmic choices they wish their simulation to tune over, and the strategy with which they wish to tune, as well as add particles to their container.

During the simulation, AutoPas provides the user with several ways in which they can interact with their particles.

- `begin` provides the user with an iterator, with which they can loop over all particles, which can be used, for example, for position or velocity updates.
- `getRegionIterator` provides the user with an iterator over a specific region of space, which can be used, for example, for boundary condition handling.
- `iteratePairwise` applies the chosen functor to all relevant pairs of particles, with any tuning being handled internally in this function.

AutoPas is only designed as a node-level library, and, as such, any distributed-memory parallelism must be implemented by the user. As discussed by Seckler et al. [7], the user can decompose the domain into several spatial regions, each handled by a separate MPI-rank and a separate AutoPas object. The result of such an implementation is that different domain regions can use different algorithms.

AutoPas' Molecular Dynamics example code, *md-flexible*, demonstrates how the above interfaces can be used to write an MD simulator. In particular, it demonstrates the use of AutoPas across multiple MPI-ranks, for different boundary conditions, and more.

4.2. Tuning strategies

In this paper, we present AutoPas' application to multi-site MD as a proof-of-concept. Therefore, we will focus on the default tuning strategy, called *Full Search*.

In the Full Search strategy, over the course of several time steps, and thus several calls to `iteratePairwise`, every algorithmic choice is trialled for a few timesteps in a row, with the time taken for each iteration recorded. For every algorithmic choice, a weighted average time is computed, which balances the average time spent building the Particle Container with the average time spent not, in accordance with the rebuild frequency.

As each algorithmic choice results in no difference in accuracy, beyond machine precision, force calculations made during tuning are still used to progress the simulation — and so the only disadvantage of tuning comes from needing to test suboptimal implementations.

5. Implementing multi-site molecular dynamics with AutoPas

There are several ways to implement Multi-Site forces for Molecular Dynamics. These range from integrators, such as RATTLE, which calculate positions and velocities of sites in a constrained manner in keeping with the molecular model [18], to modelling molecules as rigid bodies with fixed relative positions, such as for simulators like ls1-mardyn, which results in far smaller memory usage.

In this paper, we focus on the latter technique. Here, molecules are represented by a centre-of-mass position, velocity, force, quaternion, angular velocity, torque, and molecule type ID. During a force calculation, relative site positions for the molecule's type are looked up, translated by the centre-of-mass position and rotated by using the quaternion.

To reduce the number of distance calculation checks, we use a cutoff criterion based on centre-of-mass to centre-of-mass distances and apply it to all site-site interactions for a given pair of molecules.

Finally, actual site positions can be updated by updating the centre-of-mass position and velocity with a standard velocity verlet or similar algorithm, and quaternion and angular velocity using a rotational velocity verlet algorithm such as presented by Rozmanov et al. [19].

Constructing site positions on-the-fly will result in computational overhead, particularly for many-site molecules. The advantage to this, however, is a far smaller storage space required per molecule, as well as fewer load operations on the memory channels.

A new multi-site molecule class and complementary multi-site Lennard-Jones force functor were developed, as part of AutoPas' Molecular Dynamics library of example particle and functor classes, implementing the above scheme.

For implementations optimised for vectorisation, further care must be taken. In this paper, the focus is on the algorithmic properties of the methods, with hardware-specific vectorised implementations the subject of ongoing work.

6. Comparison of methods for short-range multi-site molecular dynamics

In previous works [7], it has been shown that the algorithmic choice varies based on factors such as density and homogeneity. In this paper, we shall show how the best algorithm varies with the number of sites, for different molecular distributions, hardware, and numbers of threads.

We compared Linked Cells, Verlet Lists, and Pairwise Verlet Lists, with the traversals discussed in Section 2.2, across two very different distributions of molecules: a uniform distribution and a highly gaussian distribution. Both distributions were spread across a $100 \times 100 \times 100$ domain, with periodic boundary conditions. A cutoff of $r_c = 3$ and a verlet skin of $\Delta_s = 0.667$ were used. The choice of verlet skin is partially arbitrary, as this depends heavily on factors such as the speed of molecules and the step-size, and so we make no global statement comparing Verlet List approaches to Linked Cells in all scenarios. This particular verlet skin length was found to provide an easily presentable demonstration of the trends investigated in this paper. Furthermore, so that the Linked Cells are rebuilt at the same frequency as the Verlet-based approaches, a cell size of $r_c + \Delta_s$ is used. Thus, both cells methods have dimensions of $3.667 \times 3.667 \times 3.667$, amounting to an average of 24.7 molecules per cell. For academic purposes, in this paper, we will consider implementations of C08 and C18 with and without using N3L.

The methods were compared for molecular models ranging from 1 to 10 Lennard-Jones sites. All methods and site numbers were run for 10 iterations, including one in which the container is set up, and the average wall time was taken.

The experiments in this section were carried out using AutoPas' MD test bed, md-flexible using a personal workstation and a node of Fugaku, the TOP2 supercomputer hosted at RIKEN, Japan.

The workstation was a Dell Precision 3650 Tower with an Intel Core i7-10700 CPU clocked at 2.9 GHz with 8 cores, each with 2 threads per core for 16 total, and a memory bandwidth of 45.8 GB/s.[2] Each node of Fugaku has an A64FX CPU clocked at 2.0 GHz with 48 compute cores and a memory bandwidth of 1024 GB/s.[3] Both experiments presented in this section were carried out with the workstation with 16 threads and a node of Fugaku with both 16 and 48 threads.

In Fig. 3, we present the results from these experiments.

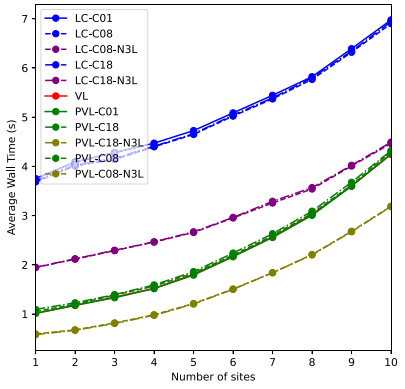
6.1. Uniform molecule distribution

We simulated 5×10^5 molecules distributed randomly with a uniform distribution. In Fig. 3 (left), we present the results from these experiments.

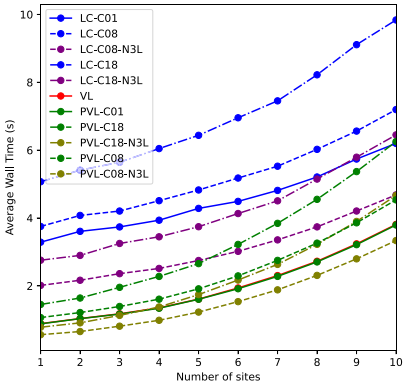
We note, firstly, that normal and Pairwise Verlet Lists outperform Linked Cells. Such a result, in itself, is not of significance, however, it is worth pointing out that, for both the N3L and non-N3L methods, this out-performance is by an approximately constant margin. The time for molecule-to-molecule interactions can roughly be split into two sections: time for cutoff criterion calculations and time for force calculations. As the number of sites increases, so too does the cost of the force calculation — however, as this time is independent of the particle container, the total time difference between the containers remains constant, at least for homogeneous scenarios. This suggests that for large molecular models, the choice of particle container becomes of limited relevance in homogeneous scenarios.

[2] https://ark.intel.com/content/www/us/en/ark/products/199316/intel-core-i710700-processor-16m-cache-up-to-4-80-ghz.html

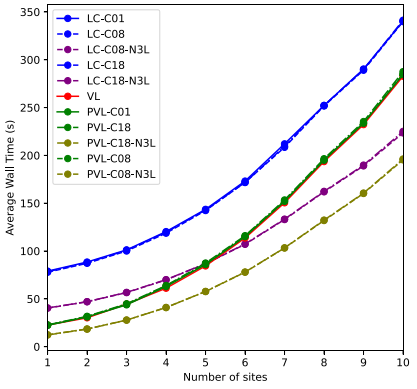
[3] https://www.fujitsu.com/global/about/innovation/fugaku/specifications/



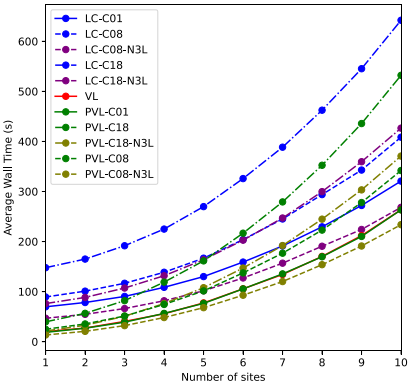
(a) Uniform - Workstation - 16 Threads



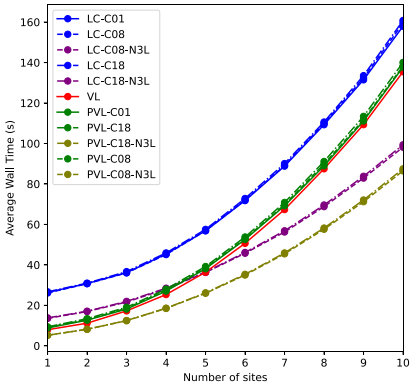
(b) Gaussian - Workstation - 16 Threads



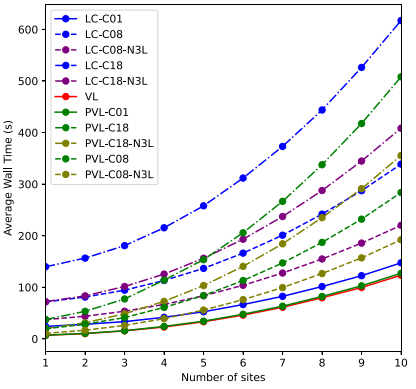
(c) Uniform - Fugaku - 16 Threads



(d) Gaussian - Fugaku - 16 Threads



(e) Uniform - Fugaku - 48 Threads



(f) Gaussian - Fugaku - 48 Threads

Fig. 3. Comparison of methods for a uniform distribution of 5×10^5 molecules (left) and a Gaussian distribution of 10^5 molecules with centre $(50, 50, 50)$ and standard deviation $(10, 10, 10)$ (right), both on a $100 \times 100 \times 100$ domain, with periodic boundary conditions. Methods were compared for molecular models with 1 to 10 sites. Experiments were run on an Intel workstation with 16 threads (top), an A64FX compute node of Fugaku with 16 threads (middle), and an A64FX compute node of Fugaku with 48 threads (bottom). For the uniform scenario, the performance of different traversals of the same particle container do not vary significantly, so the lines cover each other. For all graphs, the performance of VL and PVL-C01 closely matches, and so frequently the line for VL is covered by the line for PVL-C01.

Secondly, the traversal has no noticeable difference in the performance of the method. This is understandable for a non-sparse homogeneous scenario — any scheduling overhead is shadowed by the density in combination with the cost of interactions, and any synchronisation wait times are minimal with such a homogeneous scenario.

Thirdly, there is a clear advantage for using N3L optimisations, which result in a little over half the computational cost for force calculations compared to their non-N3L equivalents. The relative improvement from the use of N3L does, however, decline as the number of sites increases. Exactly why this decline occurs is the subject of ongoing research, however, there still remains a significant advantage even for 10-site molecules.

Verlet Lists perform relatively better compared to Linked Cells on the Workstation than on Fugaku for all site counts. As Verlet Lists, in this scenario, result in fewer molecular interactions, they thus require less memory accesses. Fugaku's processor has a far greater memory bandwidth than the Intel processor of the Workstation, and thus the consequences of the greater memory accesses needed by Linked Cells are more minimal.

Finally, Verlet Lists and non-N3L Pairwise Verlet Lists perform very similarly. The key advantage of Pairwise Verlet Lists over Verlet Lists is a closer data locality. This suggests that the advantage of this closer data locality for these naive unvectorised force calculations are minimal.

6.2. Gaussian molecule distribution

We simulated 10^5 molecules distributed with a Gaussian distribution about the centre of the domain with a standard deviation of 10 in all dimensions. The results are shown in Fig. 3 (right).

The first key difference to the homogeneous scenario is that the traversals make an impact on performance independent of the ability to use N3L. This can be seen for both Linked Cells and Pairwise Verlet Lists for all three experiments, where C01 outperforms C08, which outperforms C18. With such a heterogeneous scenario, and thus a large load imbalance between chunks of work, any synchronisation point results in the significant idling of processors, with the 18 synchronisation points of C18 resulting in a worse performance than the 8 synchronisation points of C08.

Similarly to the homogeneous case, N3L results in almost double the speedup compared to the non-N3L variants. In both 16 thread experiments, the benefits of N3L outweigh the cost of thread idling for PVL-C08, resulting in a greater performance than any other method. In the 48 thread experiment, however, this is not the case. With a greater number of threads, and thus a greater number of threads that may be idling at synchronisation points, the synchronisation wait times of C08 outweigh the benefit of N3L.

The relative performance difference between traversals for a given container remains constant as the number of sites increases. As the distribution is the same for all numbers of sites, increasing the cost of the force calculation increases only the amplitude of the load imbalances. As such, thread idling wall time increases, however, the pattern in thread idling remains the same, and so the relative performance between traversals remains constant.

Similarly to the homogeneous scenario, the performance gain of Pairwise Verlet Lists over Linked Cell equivalents is greater on the workstation than on Fugaku. Again, this is thus explained by the greater memory bandwidth on Fugaku limiting the cost of loading memory for the molecules which do not satisfy the cutoff criterion.

6.3. Summary

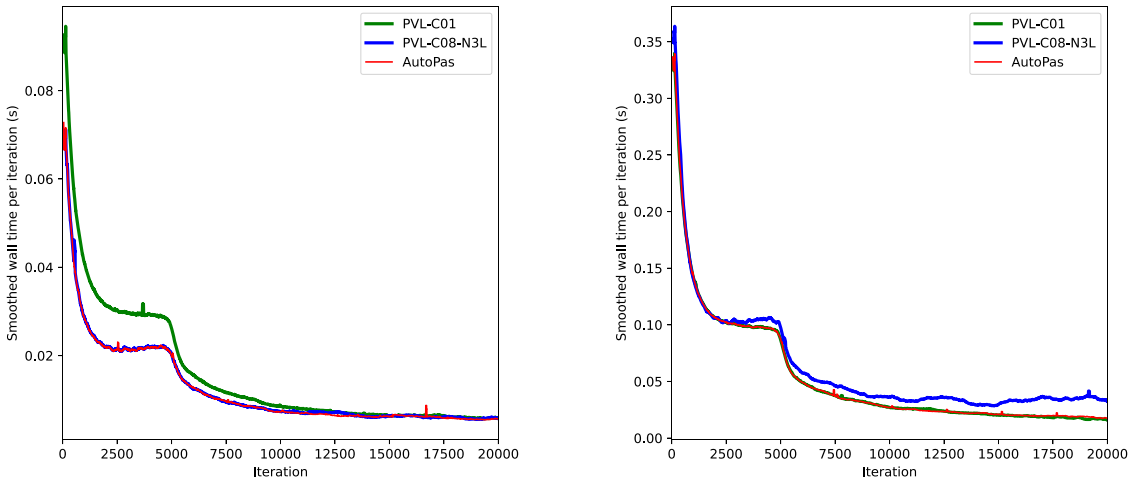
For homogeneous scenarios, we demonstrated that there is an approximately constant difference in the force calculation time for different Particle Containers, and as such a decrease in relative difference in the performance of Particle Containers. Furthermore, we demonstrated that there is no significant difference in performance between traversals. These results suggest that for large numbers of sites, and a homogeneous scenario, the difference between the methods described in this paper, with the exception of N3L, becomes minimal. As such, the choice of optimal algorithm does not change significantly for homogeneous scenarios as the number of sites varies.

On the other hand, for highly heterogeneous scenarios, we have demonstrated that relative differences in the performance of traversals remain approximately constant. These results suggest that, in heterogeneous scenarios, the optimal choice of parallel traversal becomes ever more important as the number of sites increases. Whilst choosing the optimal traversal becomes more important, the choice itself does not change with the number of sites. The only factor which does change, however, is when comparing a traversal with N3L to a traversal without N3L.

In both homogeneous and heterogeneous scenarios, the only difference in choice between algorithms as the number of sites changes is whether it is better to use a highly parallelisable traversal without N3L versus a less parallelisable traversal with N3L.

7. Auto-tuning multi-site MD simulations with AutoPas

We demonstrate the value of autotuning MD simulations using a simple test case of a tightly packed liquid exploding into an otherwise empty domain, similar to what has been used in previous works with AutoPas [7]. The simulation begins with a thin $30 \times 12 \times 30$ slice of 15540 densely packed molecules in the middle of a $30 \times 120 \times 30$ domain with periodic boundary conditions on the x- and z-axes, and no boundary condition on the y-axis. The densely packed molecules then



(a) Single Site (b) 10 sites

Fig. 4. Smoothed times per iteration for the evaluation of the pairwise force calculation in the Exploding Liquid Simulation. Times are smoothed by taking the mean time for the 50 previous iterations, or as many previous values that are available — whichever is fewer. Included, for both the 1-site case and the 10-site case, is the values from using Pairwise Verlet Lists C01 (PVL-C01 (only, Pairwise Verlet Lists C08 with Newton's 3rd Law optimisation (PVL-C08-N3L) only, and AutoPas tuning between the two.

explode into the far wider domain, filling it, before the majority of molecules pass beyond the y-axis boundary, and what remains is a sparsely filled domain. The simulation is run for 20000 timesteps with a step-size of 0.00182367.

We consider two separate simulations. In one, we have single-site molecules, and in the other, we have 10-site molecules, both with total masses of 1.0. In the single-site case, site interactions are determined by the Lennard-Jones 12-6 potential with $\epsilon = 1.0$ and $\sigma = 1.0$. To isolate the effects related to more expensive molecule-to-molecule interaction, the 10-site model is identical to the 1-site model, with 10 Lennard-Jones 12-6 sites located exactly at the centre-of-mass and with $\epsilon = 0.01$ and $\sigma = 1.0$ such that molecule-to-molecule interactions are identical to the 1-site case.

For each scenario, the simulation was carried out with PVL-C01, PVL-C08-N3L, and AutoPas' Full Search tuning strategy tuning between the two algorithms. The two algorithms were chosen as they were both the best-performing algorithms and they demonstrate the key algorithmic decision between a highly parallelisable traversal without N3L and a less parallelisable traversal with N3L. We use the Full Search algorithm by trialling each algorithm for 10 timesteps each and using the optimal for an interval of 2500 time steps.

In all cases, the simulation was run on the workstation as described in Section 6.

In Fig. 4, we see the smoothed wall time per iteration for both experiments. In the single-site case, PVL-C08-N3L outperforms PVL-C01 until approximately iteration 12500, at which point the difference between the two algorithms is negligible. In the 10-site case, the difference between both algorithms is insignificant, to begin with, before PVL-C01 outperforms PVL-C08-N3L early into the simulation.

In the single-site case, AutoPas selects PVL-C08-N3L until iteration 15140, albeit the switch here is of little advantage. In the 10-site case, AutoPas selects PVL-C01 throughout the simulation.

As shown in Section 6, good load balancing becomes important for large numbers of sites in heterogeneous scenarios. As such, the load balancing benefits of C01 generally outweigh the N3L optimisation advantages of C08 in the 10-site case, but the other way around in the 1-site case.

In the single-site case, AutoPas selects PVL-C08-N3L until iteration 15140, albeit the switch here is of little advantage. In the 10-site case, AutoPas selects PVL-C01 throughout the simulation. In these particular scenarios, there is no significant benefit to dynamically auto-tuning, however, these results demonstrate that as a minimum AutoPas can be used to select an optimal algorithm for a simulation without the user having to have the extensive knowledge required to make this choice themselves.

8. Future work

8.1. Tuning

The results in Section 7 use a Full Search algorithm with only two algorithms, which were selected because we knew in advance that these two algorithms performed the best for these scenarios. Knowing this in advance is usually not realistic.

Meanwhile, if all methods considered in Section 6 were tested during each tuning phase, the overhead from testing poorly performing methods could result in significantly worse performance than if no dynamic tuning was used at all.

The development of smart tuning strategies is, thus, paramount in ensuring that the potential for dynamic auto-tuning in Molecular Dynamics is achieved. Ideally, such a smart tuning strategy would make decisions based on data from past performances of algorithms, knowledge of correlations between the performance of different algorithms, and knowledge of how the scenario impacts the performance of different algorithms.

Results presented in this paper have indicated that the number of sites of a molecule can have a significant impact on the optimal algorithmic choice. As such, it is clear that any tuning strategy for Molecular Dynamics simulations would have to take such information into account.

As discussed in Section 4.1, classes used with AutoPas inherit from a base particle class. This base particle class contains only a 3D position, velocity, force array, and ownership status, as well as some minor additional information for the internal handling of the particle. Therefore, as the AutoPas container does not have access to domain-specific information, it is unable to gather information such as the number of sites a molecule has. A workaround, such that AutoPas can consider this information during tuning, will thus have to be developed.

8.2. Vectorisation

Gratl et al. demonstrate that in many scenarios, vectorised implementations of the Lennard-Jones force for single-site molecules can outperform unvectorised implementations [20]. To prevent loss of performance arising from the looping of the variable numbers of sites of molecules, much more care must be taken in the design of such interactions than in purely single-site simulations. Tchipev details a few different schemes for vectorised interactions between cells of molecules, one of which was implemented in ls1-mardyn [21]. Furthermore, Tchipev demonstrated the advantage of manual intrinsics over auto-vectorisation. To be able to illustrate the full potential of applying AutoPas to Multi-Site MD Simulations, such a vectorised force calculation will have to be implemented.

9. Summary

In Section 6, we analysed the key differences in the performance of algorithms as the number of sites varied. For both homogeneous and highly heterogeneous scenarios, the key change in algorithmic choice came from switching from a highly parallelisable traversal without N3L to switching to a less parallelisable traversal with N3L. We found that the choice of Particle Container made no significant difference, and that, whilst the choice of traversal had a higher impact as the number of sites increased, the ordering of the traversals from most optimal to least did not change.

In Section 7, we showed the advantage of applying AutoPas in Multi-Site Molecular Dynamics simulations, by demonstrating that it can correctly select the optimal algorithm for two simulations that differed only by the number of sites in each molecule but for which there were different optimal algorithms. To the best of our knowledge, this is the first application of auto-tuning in the field of Multi-Site Molecular Dynamics. We hope that our results will motivate further developments in auto-tuning on this topic.

The results presented in this paper can also be extended beyond the field of Multi-Site Molecular Dynamics. For unvectorised implementations, increasing the number of sites is equivalent to increasing the cost of a site-to-site interaction quadratically. As such, we expect the trends discussed in this paper to hold similarly for increasingly expensive site-to-site force calculations.

Data availability

Data will be made available on request.

Acknowledgements

This work has been supported by the project MaST, which is funded by dtec.bw – Digitalisation and Technology Research Center of the Bundeswehr. dtec.bw is funded by the European Union – NextGenerationEU. The authors thank HPCI/Riken (project hp210268, “Massively Parallel Molecular-Continuum Flow Simulation for Process Engineering Applications”) for provision of computational resources on Fugaku.

References

[1] Stephan Deublein, Bernhard Eckl, Jürgen Stoll, Sergey V. Lishchuk, Gabriela Guevara-Carrion, Colin W. Glass, Thorsten Merker, Martin Bernreuther, Hans Hasse, Jadran Vrabec, ms2: A molecular simulation tool for thermodynamic properties, Comput. Phys. Comm. 182 (11) (2011) 2350–2367.

[2] John D. Chodera, William C. Swope, Jed W. Pitera, Ken A. Dill, Long-time protein folding dynamics from short-time molecular dynamics simulations, Multiscale Model. Simul. 5 (4) (2006) 1214–1226.

[3] Gabriela Guevara-Carrion, Jadran Vrabec, Hans Hasse, Prediction of self-diffusion coefficient and shear viscosity of water and its binary mixtures with methanol and ethanol by molecular simulation, J. Chem. Phys. 134 (7) (2011) 074508.

[4] Dennis C. Rapaport, The Art of Molecular Dynamics Simulation, Cambridge University Press, 2004.

[5] Loup Verlet, Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, Phys. Rev. 159 (1) (1967) 98.
[6] Fabio Alexander Gratl, Steffen Seckler, Hans-Joachim Bungartz, Philipp Neumann, N ways to simulate short-range particle systems: Automated algorithm selection with the node-level library AutoPas, Comput. Phys. Comm. 273 (2022) 108262.
[7] Steffen Seckler, Fabio Gratl, Matthias Heinen, Jadran Vrabec, Hans-Joachim Bungartz, Philipp Neumann, AutoPas in ls1 mardyn: Massively parallel particle simulations with node-level auto-tuning, J. Comput. Sci. 50 (2021) 101296.
[8] Stefan Eckelsbach, Jadran Vrabec, Fluid phase interface properties of acetone, oxygen, nitrogen and their binary mixtures by molecular simulation, Phys. Chem. Chem. Phys. 17 (40) (2015) 27195–27203.
[9] Wolfgang Eckhardt, Alexander Heinecke, Reinhold Bader, Matthias Brehm, Nicolay Hammer, Herbert Huber, Hans-Georg Kleinhenz, Jadran Vrabec, Hans Hasse, Martin Horsch, et al., 591 TFLOPS multi-trillion particles simulation on SuperMUC, in: International Supercomputing Conference, Springer, 2013, pp. 1–12.
[10] Nikola Tchipev, Steffen Seckler, Matthias Heinen, Jadran Vrabec, Fabio Gratl, Martin Horsch, Martin Bernreuther, Colin W. Glass, Christoph Niethammer, Nicolay Hammer, et al., TweTriS: Twenty trillion-atom simulation, Int. J. High Perform. Comput. Appl. 33 (5) (2019) 838–854.
[11] Pedro Gonnet, Pairwise verlet lists: Combining cell lists and verlet lists to improve memory locality and parallelism, J. Comput. Chem. 33 (1) (2012) 76–81.
[12] Nikola Tchipev, Amer Wafai, Colin W. Glass, Wolfgang Eckhardt, Alexander Heinecke, Hans-Joachim Bungartz, Philipp Neumann, Optimized force calculation in molecular dynamics simulations for the intel xeon phi, in: European Conference on Parallel Processing, Springer, 2015, pp. 774–785.
[13] Florian Fleissner, Timo Gaugele, Peter Eberhard, Applications of the discrete element method in mechanical engineering, Multibody Syst. Dyn. 18 (2007) 81–94.
[14] Joseph J. Monaghan, Smoothed particle hydrodynamics and its diverse applications, Annu. Rev. Fluid Mech. 44 (2012) 323–346.
[15] Pablo Gómez, Fabio Gratl, Oliver Bösing, Dario Izzo, Deterministic conjunction tracking in long-term space debris simulations, 2022, arXiv preprint arXiv:2203.06957.
[16] Hongbo Zhao, Yuxiang Huang, Zhengdao Liu, Wenzheng Liu, Zhiqi Zheng, Applications of discrete element method in the research of agricultural machinery: A review, Agriculture 11 (5) (2021) 425.
[17] J.M. Domínguez, A.J.C. Crespo, M. Gómez-Gesteira, J.C. Marongiu, Neighbour lists in smoothed particle hydrodynamics, Internat. J. Numer. Methods Fluids 67 (12) (2011) 2026–2042.
[18] Hans C. Andersen, Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations, J. Comput. Phys. 52 (1) (1983) 24–34.
[19] Dmitri Rozmanov, Peter G. Kusalik, Robust rotational-velocity-Verlet integration methods, Phys. Rev. E 81 (5) (2010) 056706.
[20] Fabio Alexander Gratl, Steffen Seckler, Nikola Tchipev, Hans-Joachim Bungartz, Philipp Neumann, Autopas: Auto-tuning for particle simulations, in: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW, IEEE, 2019, pp. 748–757.
[21] Nikola Plamenov Tchipev, Algorithmic and Implementational Optimizations of Molecular Dynamics Simulations for Process Engineering (Ph.D. thesis), Technische Universität München, 2020.