MEMORY-EFFICIENT IMPLEMENTATION OF A RIGID-BODY MULTI-CENTER MOLECULAR DYNAMICS SIMULATION FOR NANO-FLUIDICS

W. Eckhardt
Technische Universität München, Germany

Corresponding author: W. Eckhardt, Technische Universität München, Chair for Scientific Computing
85748 Garching, Boltzmannstraße 3, Germany, eckhardw@in.tum.de

Motivation. The memory footprint of molecular dynamics simulations (MD) is important on systems with accelerator hardware such as GPGPUs or on systems like BlueGene with as little as 512 MB memory per core. This holds especially for coupled molecular dynamics lattice boltzmann simulations [1] whith high memory demands. We present an implementation with low memory footprint for rigid-body multi-centered MD in the programme package ls1/Mardyn [2].

Rigid-body multi-center molecular dynamics simulation. We model a fluid as a system of N discrete particles, where each particle has multiple interaction sites which are fixed relative to the center and orientation of the molecule. All molecules i and j impact each other through pairwise interaction of their sites resulting in a force $F_{ij} = \sum_{n \in sites_i} \sum_{m \in sites_j} F_{nm}(r_{nm})$ and torque $\tau_{ij} = \sum_{n \in sites_i} \sum_{m \in sites_j} r_{nm} \times F_{nm}(r_{nm})$, where r_{nm} is the distance of the sites. A molecule type modelling a given matter may have any arbitrary number of centers of different potential types, e.g. Lennard-Jones or dipoles. Typically, for short-range potentials like the Lennard-Jones potential a cutoff radius r_c is introduced and the contributions of distant particles is neglected. This can be efficiently implemented with the Linked-Cell algorithm, where the computational domain is subdivided in cells of length r_c. In order to find the neighbours of a given molecule, only the cell of the molecule and its 26 adjacent cells have to be searched, reducing the overall complexity of the force calculation from $O(n^2)$ to $O(n)$. As most memory is dedicated to the storage of particle data, we developed and evaluated two implementation alternatives and combined them to gain memory efficiency while maintaining good performance.

		Argon (1CLJ)	Ethylenoxide (3CLJD)	Benzene (6CLJ6Q)
Alt. A	SP	192	276	528
	DP	292	460	964
Alt. B	SP	88	88	88
	DP	164	164	164

Table 1: Memory requirement in Bytes per molecule for different test fluids.

Alternative A. Similar to the fly-weight design pattern, each molecule can be described by position, velocity, orientation, angular velocity, an id, and a flag indicating the type of the molecule, i.e. its interaction centers. Additionally, force and torque have to be stored, resulting in a memory consumption of 88 Bytes (single precision) or 164 Bytes (double precision) per molecule. During the interaction calculation the force and torque are accumulated for the molecule instantaniously. However, for each site-site interaction, the positions of the sites have to be calculated from the center and orientation of the molecule, causing a considerably larger runtime.
These multiple redundant calculations are avoided by **alternative B**. For each molecule the positions of its interaction sites as well as for oriented sites like dipoles the orientations are calculated one and stored explicitely at the beginning of an iteration. During the calculation of the interactions, the force is being accumulated for each site seperately, and only at the end of the force calculation the total force effective on the molecule as well as the torque is calculated. As the number of interaction sites per molecule is not known at compile time, the addidional temporary memory has to be allocated dynamically, so each molecule consumes memory for pointers as well as the memory allocated if needed, but the runtime benefits.

Hybrid Scheme. We combined both approaches to form a memory efficient implementation at even slightly better performance. Initially, we store all molecules as described as alternative A. When a cell is first searched for interacting particles, the molecules of that cell are converted into representation B (marked as AB). Similarily, when a cell is searched for the last time, its molecules are converted back to the memory efficient representation A (marked as BA). This preserves the good performance of scheme B at low memory footprint, which is now quasi-independent of the material being simulated. Moreover any kind of data reordering fits naturally into the pattern suggested opening the doorway to efficient implementations such as vectorization.

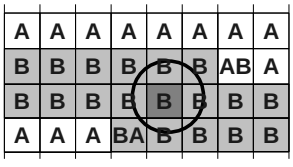

Figure 1: Molecule representation during the cell traversal.

[1] Neumann P., and Hoffman P., and Eckhardt W., and Harting J.: *All Good Things Come in Threes: Current Software and Algorithmic Developments for Molecular Dynamics-Lattice Boltzmann Simulations*. Poster; Cecam Workshop 2011: Multiscale Modelling of Simple and Complex Liquid Flow Using Particle-Continuum Hybrids. http://www5.in.tum.de/pub/neumann_cecam2011.pdf.

[2] Buchholz, M. and Bungartz, H.-J. and Vrabec, J: *Software design for a highly parallel molecular dynamics simulation framework in chemical engineering*. In: Journal of Computational Science, 2 (2011), 124–129.