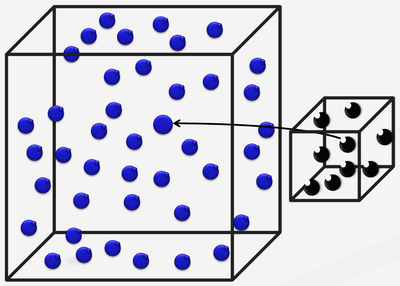
A Massively Parallel Implementation for the Grand Canonical Monte Carlo Simulation

Eyad Hailat[a], Jason Mick[b], Kamel Rushaidat[a], Loren Schwiebert[a], and Jeffrey Potoff[b]

Departments of Computer Science[a] and Chemical Engineering[b], Wayne State University, Detroit, MI

Overview

The Graphics Processing Units (GPUs) have been of interest for scientific applications lately due to the massive parallelism that they provide compared to multicore devices. However, different algorithmic designs are required to use the GPU resources and hide parallel overhead. In this research we compare results obtained from sequential and parallel implementations of Monte Carlo Grand Canonical ensembles for the Lennard-Jones Particles, and enhancing the performance of the parallel code using a cell list algorithm.



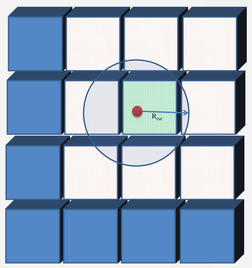
The simulation box and reservoir

Background

The grand canonical method has two types of moves. Atoms may move inside the box , or between the box and the reservoir, one move attempt at a time.

The grand canonical method can be applied to problems such as:

- Simulate adsorption Isotherms.
- Be used in numerical simulations to accurately predict properties of materials and their guest-adsorption characteristics.
- Determine the equation of state of the LJ fluid.

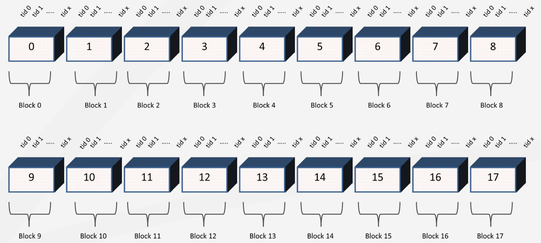


The simulation box can be decomposed into smaller domains, called cells.

Method

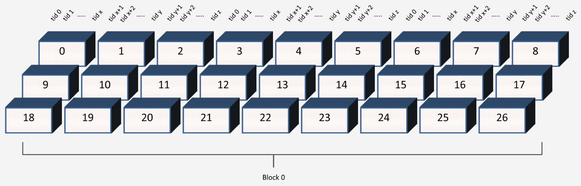
We compare results obtained from sequential and parallel implementations of Monte Carlo Grand Canonical ensembles for the Lennard-Jones Particles, and enhancing the performance of the parallel code using a cell list algorithm.

We applied cell lists to run entirely on the GPU. 26 adjacent cells for each cell is constructed. 1, 3, 9, or 27 cells handled by a block.



Each thread block handles 1 cell (27x1).

Each thread block handles 9 cell (3x9).



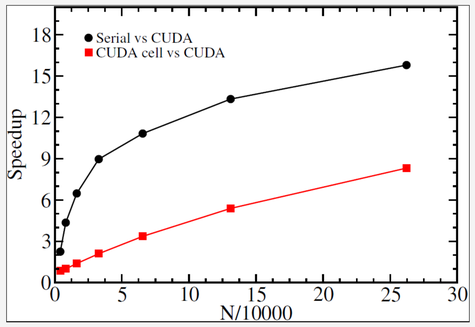
One thread block handles 27 cells(1x27).

Results

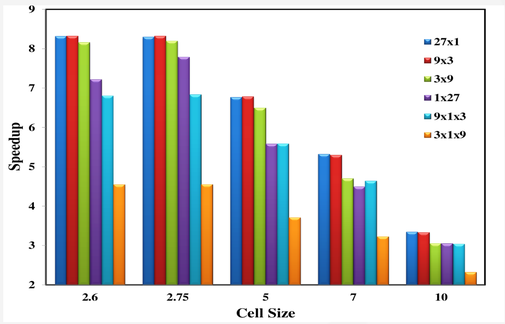
All experiments have been executed using a GeForce GTX 480 card, and a PC with an Intel Core i5-2500k CPU loaded with 8 GB of RAM.

NVIDIA GeForce GTX 480 (Image courtesy of hwigroup.net).

All measurements have used one million simulation steps (attempts) and particle radius cut 2.5. The number of particles in the box ranging from 512 to 262,144 with a corresponding volume from 853.3 to 436,905.6, doubling as the number of particles doubles.

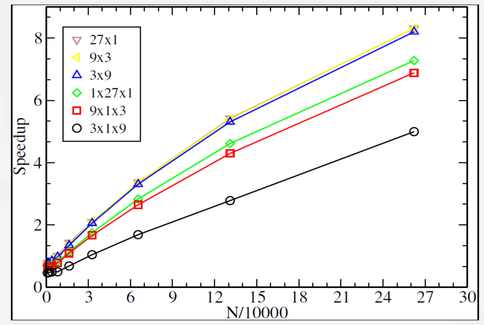


Speedup of 3 algorithms: Serial vs. original CUDA and original CUDA vs. CUDA with cell list.



Different cell sizes have different speedup for 262,144 particles and volume 436,905.6.

A fifteen times speedup has been achieved for the CUDA code over a single core code, and an additional factor of eight times speedup for the use of a CUDA implementation that has cell lists over the original CUDA code without the cell list for a problem of size 262,144 particles running on a commodity desktop loaded with a commodity GPU.



Different algorithm implementations show different speedup.

Conclusion

We present a very large scale parallel implementation fully implemented on the GPU for the grand canonical ensemble method.

Although the average total number of threads for the kernel call 27x1 and 9x3 is the same, the latter case is better in managing GPU resources. Also, for short range particles in our model, the smaller the cell size the better. As a result, we would recommend the use of 9x3 algorithm with a cell of size 2.75 for this problem under study.

Acknowledgments

This work has been supported by Wayne State University's Research Enhancement Program (REP) and National Science Foundation (NSF) grants NSF CBET - 0730768 and OCI - 1148168. Also, we thank NVIDIA for donating some of the graphics cards we used in this study.