ARTICLE IN PRESS

Computer Physics Communications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computer Physics Communications

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

GPU-accelerated Gibbs ensemble Monte Carlo simulations of Lennard-Jonesium

Jason Mick[a], Eyad Hailat[b], Vincent Russo[b], Kamel Rushaidat[b], Loren Schwiebert[b], Jeffrey Potoff[a,*]

[a] Department of Chemical Engineering and Materials Science, College of Engineering, Wayne State University, Detroit, MI 48201, USA

[b] Department of Computer Science, College of Engineering, Wayne State University, USA

ARTICLE INFO

Article history:
Received 23 October 2012
Received in revised form
7 June 2013
Accepted 25 June 2013
Available online xxxx

Keywords:
Monte Carlo
Canonical ensemble
Gibbs ensemble
GPU
CUDA

ABSTRACT

This work describes an implementation of canonical and Gibbs ensemble Monte Carlo simulations on graphics processing units (GPUs). The pair-wise energy calculations, which consume the majority of the computational effort, are parallelized using the energetic decomposition algorithm. While energetic decomposition is relatively inefficient for traditional CPU-bound codes, the algorithm is ideally suited to the architecture of the GPU. The performance of the CPU and GPU codes are assessed for a variety of CPU and GPU combinations for systems containing between 512 and 131,072 particles. For a system of 131,072 particles, the GPU-enabled canonical and Gibbs ensemble codes were 10.3 and 29.1 times faster (GTX 480 GPU vs. i5-2500K CPU), respectively, than an optimized serial CPU-bound code. Due to overhead from memory transfers from system RAM to the GPU, the CPU code was slightly faster than the GPU code for simulations containing less than 600 particles. The critical temperature $T_c^* = 1.312(2)$ and density $\rho_c^* = 0.316(3)$ were determined for the tail corrected Lennard-Jones potential from simulations of 10,000 particle systems, and found to be in exact agreement with prior mixed field finite-size scaling calculations [J.J. Potoff, A.Z. Panagiotopoulos, J. Chem. Phys. 109 (1998) 10914].

1. Introduction

Over the last 50 years, Monte Carlo simulations have been used extensively to provide fundamental insight regarding the relationship between atomic-level interactions and macro-scale phenomena. In the late 1980s through the 1990s, a proliferation of new algorithms, including Gibbs ensemble [1,2], configurational-bias [3–7], Gibbs–Duhem integration [8], and histogram-reweighting techniques [9–12] simplified greatly the determination of vapor–liquid coexistence curves. The Gibbs ensemble method is particularly versatile; in addition to vapor–liquid coexistence, this method has been used to determine adsorption equilibria in single [13–16] and multi-component systems [17,18], membrane equilibria, solid–fluid [19] and solid–vapor [20,21] equilibria. Unlike molecular dynamics, where highly optimized parallel codes are widely available to the public [22–26], most Gibbs ensemble Monte Carlo calculations (GEMC) have been performed with codes that execute on a single CPU core. This limits calculations to relatively small system sizes (<5000 atoms), and precludes the use of GEMC for the simulation of equilibria in most biological systems.

There have been numerous attempts to parallelize Monte Carlo simulations based on three core algorithms, *embarrassingly parallel*, *farm*, and *domain decomposition*. The *embarrassingly parallel* approach is the simplest to implement; a replica of the system is placed on each processor core, and each instance runs independently without any communication between processors. For systems with short equilibration periods, parallel efficiencies near 100% are possible. However, for systems with long equilibration periods, this method quickly becomes inefficient. In the *farm* algorithm [27], also known as energetic decomposition, the energy calculation is distributed over the available processing cores. Some efforts have shown improved performance with this method [28], but the farm algorithm has been avoided due to communications overhead noted in early comparative studies between energetic decomposition and embarrassingly parallel algorithms [29]. *Domain decomposition* is an example of a modified Markov chain algorithm [30], and leverages the fact that in large systems, particles outside the range of interactions of other particles may be moved simultaneously. The simulation box is split into cells, and interactions on each cell are calculated on a single CPU core. The amount of inter-processor communication is reduced compared to the farm method. In early work, domain decomposition was dismissed as

* Corresponding author.
E-mail address: jpotoff@wayne.edu (J. Potoff).

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.06.020

inefficient, since its performance was worse than the embarrassingly parallel method [31]. More recently, however, it has been shown that by using a sequential updating algorithm the parallel efficiency of domain decomposition may be improved significantly [32–34]. Additional modified Markov chain algorithms have been proposed where displacement of all particles, or a subset of particles, is attempted simultaneously [35]. These methods are referred to as hybrid Monte Carlo methods, due to the use of velocities to determine new trial locations, which are typically used in Newtonian-physics based molecular dynamics simulations. The efficiency of hybrid Monte Carlo methods tends to be dictated by the phase [36] and type of atoms or molecules studied [37].

In addition to general methods of parallelizing atomistic Monte Carlo simulations, some specialized ensemble-specific techniques have been demonstrated. Chen and Hirtzel proposed a macro state Markov chain model (MSMCM) for parallelization of simulations in the grand canonical ensemble [38–40]. Spatial updating algorithms, combined with sequential updating and domain decomposition, have been used to improve the efficiency of parallel Monte Carlo simulations in the grand canonical ensemble [34]. Parallel grand canonical Monte Carlo simulations have been combined with molecular dynamics for the simulation of diffusion in porous materials [41,42]. Additional parallelization efforts have been focused on the configurational-bias algorithm, which is crucial to achieving reasonable acceptance rates for molecule transfers in grand canonical and Gibbs ensemble Monte Carlo simulations. In order to improve the success rate for growing chain molecules in a dense system, Esselink et al. proposed evaluating multiple trial locations for the first bead in parallel [43]. This methodology for particle exchange was combined with hybrid Monte Carlo moves for particle displacement in Gibbs ensemble Monte Carlo where a parallel efficiency of approximately 80% was observed [36,43].

In recent years the field of parallel computing has shifted its focus from the CPU to a new kind of hardware, the graphics processing unit (GPU), which was originally created satisfy the demand for increasing realism in virtual video game environments [44]. GPUs have outpaced CPU in terms of units of computational power per electrical power consumption and computational power per discrete hardware costs. GPUs typically devote more transistors to fast parallel math processing than CPU at the expense of flexibility. Compared to the relatively small collection of complex processing cores found in modern CPU, GPUs are composed of a collection of simpler processors (streaming multiprocessors – "SMs" – each composed of dozens of shader core subunits) capable of massively parallel math operations. GPUs have several types of volatile storage available for simulation variables—DRAM, constant cache memory, global cache memory, shared memory and registers. DRAM has the highest latency, while registers offer the lowest latency. However, the DRAM bank has significantly larger capacity than the registers or cache, hence part of the challenge in developing efficient algorithms to utilize the GPUs is optimizing variable distribution and memory throughput.

Given inherent hardware differences between multi-core CPU and GPU, the creation of optimized GPU algorithms requires a fundamental reexamination of parallelization methods. Some methods that appear less viable for CPU parallelization may perform well on the GPU, and the converse may also be true. Additionally, there is an opportunity to develop new algorithms/techniques, which were not considered or postulated due to the architectural differences between single instruction multiple data (SIMD) devices and the traditional CPU-based multiple-instruction multiple-data (MIMD) clusters and supercomputers. Despite relatively mature GPU-enabled molecular dynamics simulation engines [25,45–49], there has been significantly less exploration of Monte Carlo simulations on GPU. As with molecular dynamics, GPGPU computing is expected to bring sizeable benefits to Monte Carlo simulations. Early work has focused on simple systems, due to the relative complexity of writing optimal GPU code. 2D and 3D simulations of Ising models were recently accelerated using single [50] and multiple GPUs [51]. GPU-driven canonical ensemble simulations of hard spheres [52], have also been published. Canonical ensemble simulations of methane in a zeolite framework have been performed on GPU for small systems ($N_{\text{methane}} \leq 128$ particles) [53]. This code was later expanded to perform GPU-accelerated grand canonical Monte Carlo simulations of methane and CO_2 in a zeolite [54–56].

In this work, an implementation of Gibbs ensemble Monte Carlo on the GPU is presented. A straightforward implementation of the farm algorithm is used for parallelization of the pair-wise energy calculations for the three distinct move types: particle displacement, swap and volume exchange. The choice of the farm algorithm is motivated by the fact that for complex molecules, such as those found in biomolecular simulations, long equilibration periods are required for adequate sampling of phase space, which reduces significantly the efficiency of embarrassingly parallel methods. Simulations of biomolecular systems present an additional problem of system size. While Monte Carlo simulations for phase equilibria and physical property prediction are performed typically for systems of 5000 atoms or fewer, typical biomolecular simulations contain tens to hundreds of thousands of atoms. Therefore it is important to have an implementation that scales well with system size. As a demonstration, calculations are performed for systems of Lennard-Jones beads ranging in size from 512 to 131,072 particles. While the properties of the Lennard-Jones fluid may be studied easily with a serial code, the wealth of data in the literature makes this the ideal system for the evaluation of new methodologies. Calculated energies, pressures, saturated liquid and vapor densities are presented for a range of temperatures and compared to literature data for validation. Detailed assessments of the performance of the GPU-accelerated GEMC code relative to an equivalently designed and optimized serial code are presented for a range of GPU and CPU. Additional data are presented for simulations in the canonical ensemble.

2. GPU implementation

The Gibbs ensemble methodology utilizes two simulation boxes, each representing a region of fluid deep within their respective phases (vapor, liquid) [1]. The three criteria for equilibria, equality of temperature, pressure and chemical potential between phases, are satisfied through three types of moves: particle displacement within a phase, volume exchange and particle swaps between phases. Acceptance probabilities for each move are defined as follows

$$\text{displacement: acc}\,(o \to n) = e^{-\beta[U_{\text{new}} - U_{\text{old}}]} \tag{1}$$

$$\text{volume swap: acc}\,(o \to n) = \min\left(1, \left(\frac{V_1^{\text{new}}}{V_1^{\text{old}}}\right)^{N_1+1} \left(\frac{V_2^{\text{new}}}{V_2^{\text{old}}}\right)^{N_2+1} e^{-\beta[\triangle U_1 + \triangle U_2]}\right) \tag{2}$$

$$\text{particle swap: acc (box 1} \to \text{box 2)} = \min\left(1, \frac{N_1 V_2}{(N_2+1)\,V_1} e^{-\beta[\triangle U_1 + \triangle U_2]}\right) \tag{3}$$

$$\text{particle swap: acc (box 2} \to \text{box 1)} = \min\left(1, \frac{N_2 V_1}{(N_1+1)\,V_2} e^{-\beta[\triangle U_1 + \triangle U_2]}\right) \tag{4}$$

where the subscripts 1 and 2 refer to each simulation box, $\triangle U_i = U_i^{\text{new}} - U_i^{\text{old}}$ denotes the energy change for the trial move in box i, $V = V_1 + V_2$, and $N = N_1 + N_2$.

Please cite this article in press as: J. Mick, et al., GPU-accelerated Gibbs ensemble Monte Carlo simulations of Lennard-Jonesium, Computer Physics Communications (2013), http://dx.doi.org/10.1016/j.cpc.2013.06.020

The particle displacement and swap routines require the calculation of pair-wise interactions between the selected particle and the rest of the particles in the system. For the displacement move, there are $N_{box} - 1$ evaluations of the Lennard-Jones potential for the trial location, where N_{box} is the number of particles in the simulation box of interest. The old energy can be either stored in local memory, or recalculated using $N_{box} - 1$ evaluations. The swap move requires $N_{box,in}$ total evaluations of the Lennard-Jones potential for the trial insertion site, while the old energy, as in the displacement move, can be either stored or recalculated via $N_{box,out} - 1$ pair evaluations. In the volume swap move, distances between all particles in both boxes are rescaled, which requires $\frac{N^2-N}{2}$ total evaluations of the Lennard-Jones potential per box. For small particle numbers, it may be faster to store the r^{-12} and r^{-6} components of the pair-wise Lennard-Jones interaction for each pair of beads and simply rescale the total energy of each box instead of completely recalculating the energy for each attempted volume exchange. As systems sizes increase, however, the memory requirements make such a methodology impractical on the GPU. For a 10,000 atom system, storage of the $\frac{N^2-N}{2}$ unique r^{-12} and r^{-6} components would require approximately 762 MB of memory on the GPU. Furthermore, this optimization is not possible for molecular systems, and therefore was avoided.

The architecture of the GPU is shown schematically in Fig. 1. Threads are grouped into warps, warps are grouped into blocks, multiple blocks are combined in a single streaming multiprocessor (SM), and streaming multiprocessors are combined to form the GPU. On GPU with *Fermi* architecture, there are 32 threads in a single warp. This architecture is a natural fit for the method of energetic decomposition, where parts of the energy calculation are split, or farmed, among available resources. This parallelization methodology is also well suited to problems of interest to our group, e.g. simulation of equilibria in biomolecular systems, which contain tens to hundreds of thousands of atoms. In this work, pair-wise energy calculations for the particle displacement and swap moves were performed on the GPU by calling N threads, where $N - 1$ is the number of unique pair interactions to be evaluated; each thread (except the thread whose index corresponds to the particle moved) was used to calculate the interaction between two particles. It might be tempting to use $N - 1$ threads, but testing showed that this makes both indexing the particles less intuitive and creates thread divergence (as the threads after the selected particle index would have to add an offset); hence N threads were used and a simple conditional was used to idle the test particle's thread. For the volume exchange move, energy calculations on the GPU were performed with $4N$ threads, a thread count that allowed for improved memory addressing. Energy calculations for the two simulation boxes were assigned as two individual kernel calls to separate streams, allowing the new energy of each box to be calculated simultaneously if sufficient resources are available. All calculations used 128 threads per block, which has been shown to provide efficient utilization of the GPU for these types of calculations [53]. A tree summation algorithm was used to coalesce the results from the individual threads. Within a block, the results from the first half of the threads are added to the second half of the threads. This process is repeated, with half as many threads participating in each phase, until the entire calculated results from a block is stored in the first thread's shared memory array position. As each block finishes, its first thread increments a thread-safe global counter, and copies its results from shared memory to a global array. The final block to execute then copies the partial sums from other blocks to positions in its shared memory array. It then repeats the previous data coalescing process, yielding a final inter-block sum.

Additional complexities arise in the volume move, since the pair-wise energies for all N particles in the system have to be

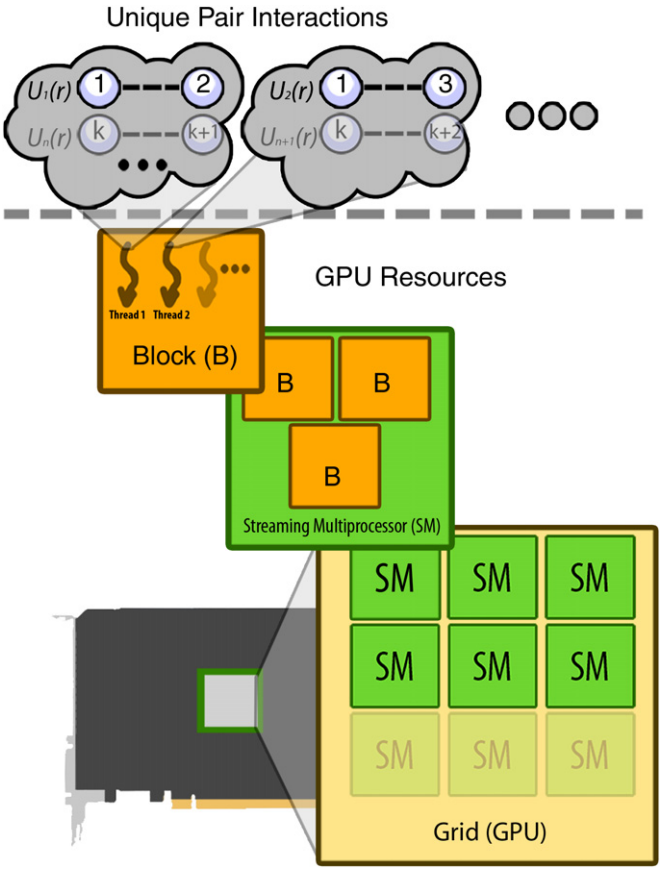

Fig. 1. In this work, energy calculations are parallelized using the method of energetic decomposition. The calculation of pair interactions is distributed among available threads. If there are more unique pairs than available threads (shown as $k, k+1, \ldots$) then threads receive another pair interaction to process, and so on, until all interactions have been farmed out. Threads are organized into "blocks" (B), which are then organized onto units of hardware known as streaming multiprocessors (SMs). Multiple SMs can be found on a graphics processing unit (GPU).

recalculated. This requires $(N^2 - N)/2$ evaluations of the Lennard-Jones potential, which in a traditional serial CPU-bound code would be an $O(N^2)$ operation. The brute force approach on the GPU would be to calculate $(N^2 - N)$ interactions and correct for double counting pair-wise interactions by dividing the resulting energy by 2. This naïve approach offers easier thread indexing, but wastes resources, since $(N^2 - N)/2$ threads are performing redundant calculations. As N grows large, the device typically does not have enough hardware to waste on duplicate calculations. Hence, it is desirable to calculate only unique interactions.

A remapping algorithm was developed to allocate pair interactions to arbitrary threads, which allows direct indexing in a thread-coherent manner (the naïve alternative – using a loop to calculate the pair index – creates thread divergence). This algorithm maps the thread index to a 2-dimensional array of particle indices, which is shown graphically in Fig. 2. Interactions on the grid above the line defined by $i = j$ are unique. To transform this unique-pair index space from a triangle, which would require redundant calculations, to a rectangular block of unique indexes, the region below the equal-index line in the top left quadrant is mapped to the region of unique indices above the equal-index line in the lower right quadrant. By mapping all unique particle interactions to a contiguous index space it is possible to determine each pair interaction efficiently using an arbitrary thread. For larger systems, the evaluation space $(N^2 - N)/2$ is still too large to assign one pair per thread (e.g. for $N = 10{,}000$ there are approximately 50 million unique pairs), hence 4 rows of the remapped space are evaluated

Please cite this article in press as: J. Mick, et al., GPU-accelerated Gibbs ensemble Monte Carlo simulations of Lennard-Jonesium, Computer Physics Communications (2013), http://dx.doi.org/10.1016/j.cpc.2013.06.020

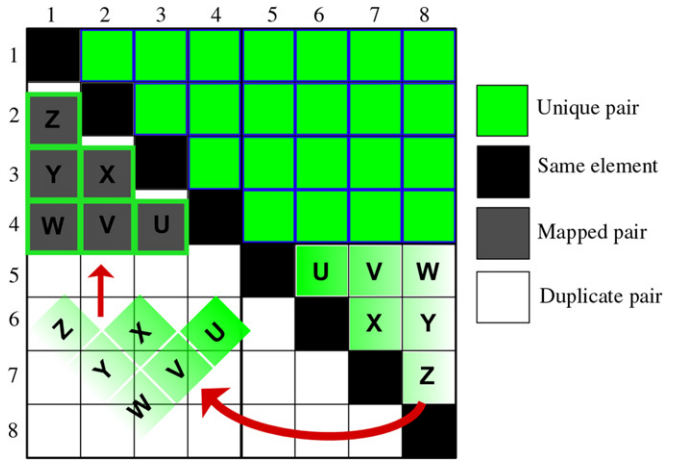

Fig. 2. Scheme for mapping of unique interaction pairs to allow for more efficient distribution of threads.

at once, which offers both effective acceleration and memory addressing gains.

The implementation presented in this work differs somewhat from the GPU implementation of grand canonical Monte Carlo by Kim et al., which uses a hybrid of energetic decomposition and embarrassingly parallel methods [54]. In the Kim method, a large number of unique Monte Carlo simulations are run on discrete blocks of the GPU, an approach similar to running independent Markov chains on individual CPU-cores; within a block pair-wise energy calculations are distributed among available threads. The hybrid methodology provides significant performance increases for the simulation of small systems sizes that are typically found in adsorption calculations, but becomes less efficient as the size of the system increases, or for systems where long equilibration periods are required.

Both the Gibbs and canonical ensemble codes were written in ANSI C. The serial version was compiled using g++ (version 4.4.6) with compiler flags -Wall- O2. The CUDA code was divided into algorithms that run on the CPU-side (host) and algorithms that run on the GPU (device). Code containing CUDA application programming interface (API) calls was compiled using the *nvcc* compiler from NVIDIA (version 4.1) with flags -fnostrict-aliasing -DUNIX -m64 -O2 -gencode=arch=compute_20,code=\ "sm_20,compute_20\".

3. Simulation details

In this work, interactions between particles are governed by the ubiquitous Lennard-Jones potential

$$U\left(r_{ij}\right) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{5}$$

where U is the configurational energy of the pair interaction, ε_{ij} is the well depth, σ_{ij} is the particle diameter and r_{ij} is the distance between particles i and j. Reduced quantities were defined as $T^* = k_B T/\varepsilon$, $\rho^* = N\sigma^3/V$, $U^* = U/\varepsilon$, $p^* = p\sigma^3/\varepsilon$.

Simulations were performed for both the truncated and tail-corrected Lennard-Jones potential. Analytical tail corrections were given by the following equations

$$U^*(r_c) = \frac{8}{9}\pi\rho^*\epsilon\sigma^3\left[\left(\frac{\sigma}{r_c}\right)^9 - 3\left(\frac{\sigma}{r_c}\right)^3\right] \tag{6}$$

$$P^*(r_c) = \frac{32}{9}\pi\rho^{*2}\epsilon\sigma^3\left[\left(\frac{\sigma}{r_c}\right)^9 - \frac{3}{2}\left(\frac{\sigma}{r_c}\right)^3\right]. \tag{7}$$

The GPU accelerated Gibbs ensemble code was validated for both the truncated and long-range corrected Lennard-Jones potential. Calculations were performed for two system sizes, $N = 500$ and $N = 10{,}000$. Initial volumes for each simulation box were determined based on the expected equilibrium density of the gas and liquid phases at the given reduced temperature, T^*. The Lennard-Jones potential cutoff was set at $r_c = 2.5\sigma$ for simulations using a truncated potential; long-range corrected data were obtained from simulations with $r_c = 3.0\sigma$. The 500 particle system was equilibrated for 1.0×10^8 Monte Carlo steps (MCS) and data were extracted from a 7.0×10^8 MCS production run. The 10,000 particle system was equilibrated for 2.0×10^7 MCS, and data were collected from a 1.8×10^8 MCS production run. Temperatures were selected to match published data [1,57–59].

Simulations in the canonical ensemble were performed at varying densities in the low and high density regimes for a system size of $N = 500$, with, and a cutoff of $r_c = 3.0\sigma$. The system was allowed equilibrate for 5.0×10^7 Monte Carlo steps (MCS) and production data was gathered from a 2.5×10^8 MCS production run. Isotherms were produced for $T^* = 0.85$ and 0.90. These simulation parameters followed those suggested by the National Institute of Standards and Technology for its benchmark data of a pure Lennard-Jones fluid [60].

Timing data for serial simulations were run on an Intel® Core™ i5-2500K CPU operating at 3.3 GHz and an Intel® Core™ 2 Quad CPU Q6600, clocked at 2.40 GHz, to ascertain the effect of CPU on the performance of the serial code. The serial and GPU codes were designed to perform an identical number of evaluations of inter-particle distances and the Lennard-Jones potential for each Monte Carlo move for a given system size. This provides a direct and unambiguous measure of the performance improvements possible using the GPU. Calculations in both the GPU and the serial code were performed using double precision variables for the storage of particle coordinates, distances, and energies. Timing data for GPU-accelerated simulations were determined for three different GPU models—the NVIDIA® GeForce™ GTX 465 (352 CUDA cores @ 1250 MHz), the GeForce™ GTX 480 (480 CUDA cores @ 1401 MHz), and the NVIDIA® GeForce™ GTX 560 Ti (384 CUDA cores @ 1701 MHz), all of which shared a common board designer, EVGA. All validation and timing results were run for five trials per data point with a randomized starting seed, to collect the average and standard deviation values for the data point.

4. Results and discussion

The GPU accelerated Gibbs ensemble Monte Carlo engine was used to determine vapor–liquid coexistence curves for the truncated and long range corrected Lennard-Jones potential for systems containing 10,000 atoms. These data are shown in Fig. 3, and show excellent agreement with the original Gibbs ensemble results of Panagiotopoulos for systems containing 500 atoms [1], as well as more modern approaches using grand canonical Monte Carlo simulations coupled with histogram reweighting methods [57–59]. Corresponding numerical data are listed in Table 1. As expected, the effect of system size is negligible at temperatures away from the critical region. The Gibbs ensemble method is well known to exhibit instabilities in the near critical region, however, this is with respect to the traditional system sizes used in GEMC calculations of 500–1000 atoms. Simulations of the 10,000 atom system were stable at temperatures up to $T^* = 1.30$, which enabled the prediction of the critical point with accuracy equivalent to histogram-reweighting calculations combined with mixed field finite-size scaling techniques. Data between $T^* = 1.2$ and 1.3 (long range corrected) or 1.0667 and 1.1696 (truncated) were fit to the density scaling law for critical temperature [61]

$$\rho_{\text{liq}} - \rho_{\text{vap}} = B\left(T - T_c\right)^{\beta} \tag{8}$$

Table 1
Data from CUDA-Gibbs ensemble Monte Carlo simulations for truncated Lennard-Jones potential with $r_{cut} = 2.5\sigma$ and the tail-corrected Lennard-Jones potential, with $r_{cut} = 3.0\sigma$. Number in parenthesis represents the uncertainty in the last digit.

T^*	ρ^*_{vapor}	ρ^*_{liquid}	U^*_{vapor}	U^*_{liquid}	P^*
Truncated Lennard-Jones potential, $r_{cut} = 2.5\sigma$					
1.1696	0.177(3)	0.454(8)	−1.40(3)	−3.00(3)	0.12(1)
1.1494	0.144(2)	0.498(3)	−1.17(2)	−3.24(2)	0.1012(8)
1.1111	0.1098(8)	0.5554(5)	−0.92(7)	−3.58(5)	0.08098(4)
1.0667	0.0797(7)	0.5999(6)	−0.69(6)	−3.87(3)	0.0615(3)
1.0256	0.0603(4)	0.6333(5)	−0.54(4)	−4.09(4)	0.0474(2)
0.9877	0.0462(4)	0.6601(5)	−0.42(4)	−4.28(4)	0.0367(2)
0.9412	0.0335(2)	0.6907(2)	−0.31(1)	−4.51(2)	0.0266(2)
Lennard-Jones potential, $r_{cut} = 3.0\sigma$ + long range corrections					
1.30	0.202(4)	0.438(5)	−1.57(3)	−3.04(2)	0.1216(9)
1.27	0.156(2)	0.486(2)	−1.25(2)	−3.35(1)	0.1065(9)
1.25	0.135(2)	0.514(2)	−1.10(2)	−3.873(9)	0.0967(4)
1.20	0.099(2)	0.564(1)	−0.83(1)	−3.873(9)	0.0772(6)
1.15	0.0733(9)	0.6057(8)	−0.634(8)	−4.171(5)	0.0599(4)
1.11	0.0578(3)	0.6337(5)	−0.511(2)	−4.378(4)	0.0484(2)
1.00	0.0294(2)	0.7008(4)	−0.277(2)	−4.894(3)	0.0249(2)
0.90	0.0141(2)	0.7521(3)	−0.142(3)	−5.312(3)	0.0115(2)
0.75	0.0030(1)	0.8208(2)	−0.0344(5)	−5.899(2)	0.0022(1)

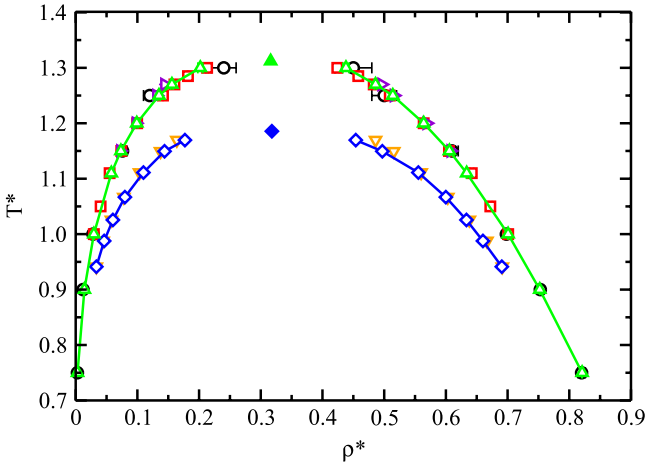

Fig. 3. Vapor–liquid coexistence curves predicted for the truncated (blue diamonds) and tail corrected (green triangles) Lennard-Jones fluid from 10,000 particle simulations in the Gibbs ensemble. Additional data shown for comparison for the tail corrected Lennard-Jones potential: Gibbs ensemble simulations for a 500 particle system (black circles) [1], grand canonical Monte Carlo simulations combined histogram reweighting (red squares) [58] and (violet triangle right) [59]. Filled symbols correspond to the predicted critical points. For the truncated Lennard-Jones potential, the data of Wilding for $r_{cut} = 2.5\sigma$ are shown (orange triangle down) [57]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

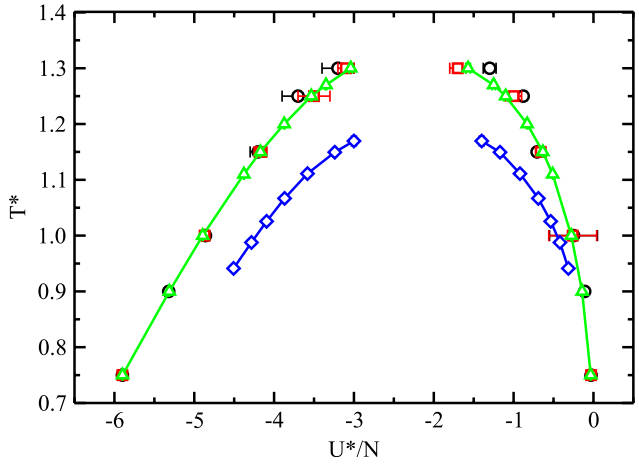

Fig. 4. Average energy per particle for vapor and liquid phases predicted from 10,000 particle Gibbs ensemble Monte Carlo simulations for the truncated, $r_c = 2.5\sigma$, (blue diamonds) and tail corrected (green triangles) Lennard-Jones potential. Data for simulations of the tail-corrected Lennard-Jones potentials performed with 300 (black circles) and 500 (red squares) particles are presented for comparison [1]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

and the law of rectilinear diameters [62]

$$\frac{\rho_{liq} + \rho_{vap}}{2} = \rho_c + A\,(T - T_c) \tag{9}$$

where $\beta = 0.325$ is the critical exponent for Ising-type fluids in three dimensions. For the tail corrected Lennard-Jones potential $T^*_c = 1.312(2)$ and $\rho^*_c = 0.316(3)$, while for the Lennard-Jones potential truncated at 2.5σ, $T^*_c = 1.186(1)$ and $\rho^*_c = 0.318(2)$. In both cases, the data are in exact agreement with prior calculations [63].

The internal energy per particle as a function of temperature for each phase is presented in Fig. 4, while the predicted vapor pressures are shown in Fig. 5. In all cases, data produced by the GPU accelerated GEMC engine are in excellent agreement with published data. These data show a significant improvement in statistical precision compared to prior calculations. This is the result of being able to run very large system sizes, which suppress fluctuations in energy and particle number, both of which scale with system size as $O(1/N^{1/2})$. Additional validation of the code was performed by running canonical Monte Carlo simulations for a variety of densities for $T^* = 0.85, 0.90$. These data may be found in the supplementary material (see the Appendix). In all cases, exact agreement was found between the predictions of the GPU-based Monte Carlo code and benchmark data from the National Institute of Standards and Technology (NIST).

Benchmarks were performed for the GPU-accelerated canonical and GEMC codes and an optimized serial code written in C++. The effects of increasing system size were examined for systems of size $N = 512$, 1024, 2048, 4096, 8192, 16,384, 32,768, 65,536, and 131,072. For simulations in the Gibbs ensemble, the volume of each simulation box was selected to produce an initial density $\rho^* = 0.3$, with particles being distributed evenly between the two boxes. Five trials of 1×10^6 Monte Carlo steps (MCS) were run for each code (serial and GPU) at each data point.

In Fig. 6, performance data are presented for the GPU-accelerated and serial canonical ensemble Monte Carlo code.

Please cite this article in press as: J. Mick, et al., GPU-accelerated Gibbs ensemble Monte Carlo simulations of Lennard-Jonesium, Computer Physics Communications (2013), http://dx.doi.org/10.1016/j.cpc.2013.06.020

Table 2
Timing data for canonical (NVT) and Gibbs ensemble Monte Carlo simulations performed on various CPU and GPU. All calculations were performed for $r_{cut} = 3.0\sigma$, with analytical tail corrections. Numbers in parentheses denote uncertainty in the last digit.

N	Q6600 (CPU)	i5-2500K (CPU)	GeForce GTX 465 + Q6600 (GPU)	GeForce GTX 480 + Q6600 (GPU)	GeForce GTX 560 + Q6600 (GPU)	GeForce GTX 560 + i5-2500K (GPU)
Canonical Monte Carlo simulations (NVT)						
512	19.827(5)	11.552(4)	13.147(7)	11.722(4)	11.000(6)	10.935(5)
1024	33.818(9)	20.987(5)	13.457(5)	11.953(7)	11.513(9)	11.465(5)
2048	61.50(1)	40.277(5)	14.48(1)	12.472(4)	12.497(9)	12.43(2)
4096	109.26(2)	73.32(2)	16.41(1)	13.827(5)	18.3(2)	18.26(3)
8192	188.54(2)	129.20(5)	26.19(2)	21.22(1)	26.10(2)	25.98(2)
16,384	350.7(2)	242.52(1)	38.03(2)	30.495(8)	42.140(8)	42.06(1)
32,768	661.6(5)	456.88(3)	68.72(1)	49.415(5)	77.332(9)	77.29(2)
65,536	1255(4)	853(3)	125.91(3)	88.715(5)	145.06(3)	144.86(9)
131,072	2683.3(4)	1690(2)	238.36(5)	163.948(9)	281.08(2)	280.75(3)
Gibbs Ensemble Monte Carlo simulations (GEMC)						
512	28.2(9)	15.9(5)	30.2(1)	24.06(8)	26.89 (4)	23.3(1)
1024	90(7)	44(5)	33.9(6)	26.9(9)	30.3(5)	27.0(3)
2048	230(20)	139(7)	39.4(2)	30.8(2)	37.1(9)	33.1(2)
4096	703(4)	417(6)	55.8(4)	43.8(9)	58.1(9)	55(1)
8192	$2.64 \times 10^3(3)$	$1.56 \times 10^3(1)$	123(1)	90(1)	136.8(8)	133(1)
16,384	$1.05 \times 10^4(5)$	$6.06 \times 10^3(3)$	349(4)	243(2)	411(2)	407(3)
32,768	$4.0 \times 10^4(1)$	$2.4 \times 10^4(1)$	$1.13 \times 10^3(1)$	$1.13 \times 10^3(1)$	$1.45 \times 10^3(1)$	$1.13 \times 10^3(1)$
65,536	$1.6 \times 10^5(1)$	$9.4 \times 10^4(2)$	$4.38 \times 10^3(4)$	$4.38 \times 10^3(4)$	$5.47 \times 10^3(4)$	$4.38 \times 10^3(4)$
131,072	$6.6 \times 10^5(9)$	$3.4 \times 10^5(4)$	$1.80 \times 10^4(1)$	$1.80 \times 10^4(1)$	$2.26 \times 10^4(1)$	$1.80 \times 10^4(1)$

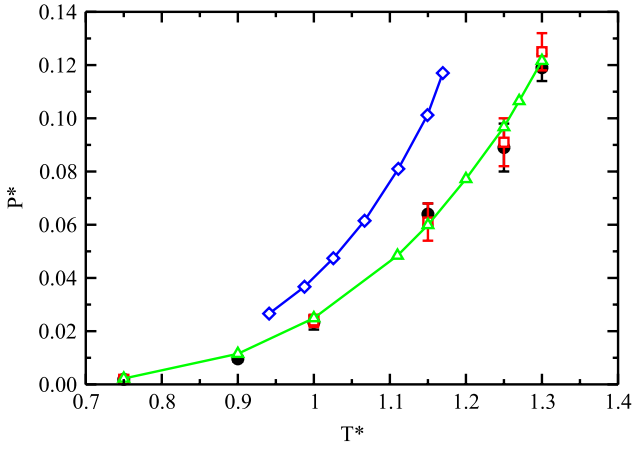

Fig. 5. Vapor pressures predicted from 10,000 particle Gibbs ensemble Monte Carlo simulations for the Lennard-Jones potential truncated at 2.5σ (blue diamonds), and the Lennard-Jones potential with analytical tail corrections (green triangles). Statistical uncertainties were smaller than the symbol size. Data from the work of Panagiotopoulos for the tail corrected Lennard-Jones potential are included for simulations performed with 300 (black circles) and 500 (red squares) particles [1]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

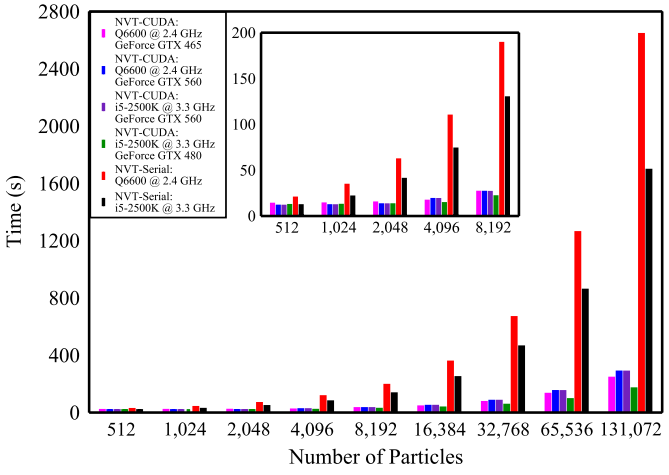

Fig. 6. Performance data for canonical Monte Carlo simulations. Timing data were determined from 1×10^6 step simulations performed at $\rho^* = 0.30$, $T^* = 1.0$ with a Lennard-Jones cutoff of 3.0σ. Inset shows a magnified view of the timing data for $512 < N < 8192$.

Tabulated data may be found in Table 2. For the largest system size, $N = 131{,}072$, the GPU-based code running on a GeForce GTX 480 (the fastest tested GPU) is approximately 10 times faster than code running on a single core of an Intel i5-2500K processor (the fastest tested CPU). The inherent overhead, due to memory transfers to and from the device over the PCI-bus, make the GPU-enabled simulation slower than the serial simulation for small particle numbers ($N < 500$). For systems between 512 and 4096 particles, there was a negligible increase execution time on the GPU, suggesting that for small system sizes memory transfers between system RAM and GPU were the primary performance bottleneck. For systems that equilibrate rapidly, Kim et al., has proposed a solution where each thread block on the GPU executes an independent Monte Carlo simulation and threads within a block are used to calculate the Lennard-Jones pair potential [53]. Their method optimizes the total throughput on the GPU, while the methodology proposed in this work is focused on minimizing the run time of a single Monte Carlo simulation. On a Tesla C2050, which has 14 SM (448 compute cores), 112 independent simulations could be launched simultaneously (14 SM × 8 blocks/SM), which produced speedups of up to 50 relative to a single core of an Intel 5530 (*Nehalem*) CPU for systems containing 512–2048 particles. While throughput was maximized, each simulation instance was running at approximately half the speed as on a single CPU core. Due to differences in computational hardware, a quantitative comparison cannot be made between this work and that of Kim et al., however, qualitatively our results show approximately 2–6 times the performance for the runtime of a single Monte Carlo instance on the GPU relative to runtime on a CPU for systems with fewer than 2000 particles, albeit at significantly reduced overall throughput.

Calculations were performed with several different GPU and CPU combinations to determine the effect of hardware on the expected performance of the code. For the serial code, the *Sandy*

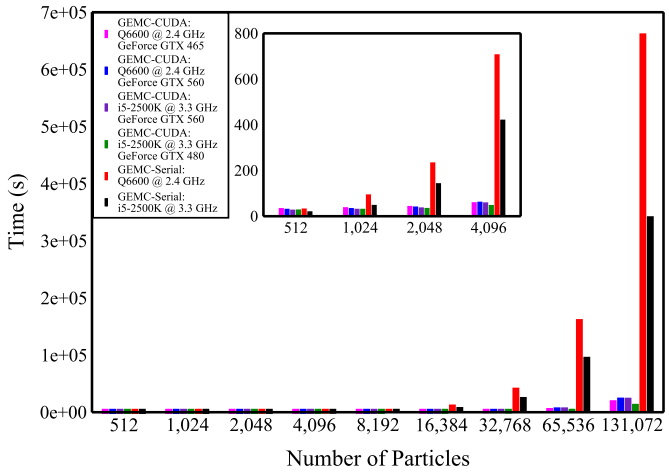

Fig. 7. Performance data for Gibbs ensemble Monte Carlo simulations of Lennard-Jones particles with $r_{cut} = 3.0\sigma$. Data correspond to a simulation of 1×10^6 Monte Carlo steps. Inset shows a magnified view for $500 < N < 4096$.

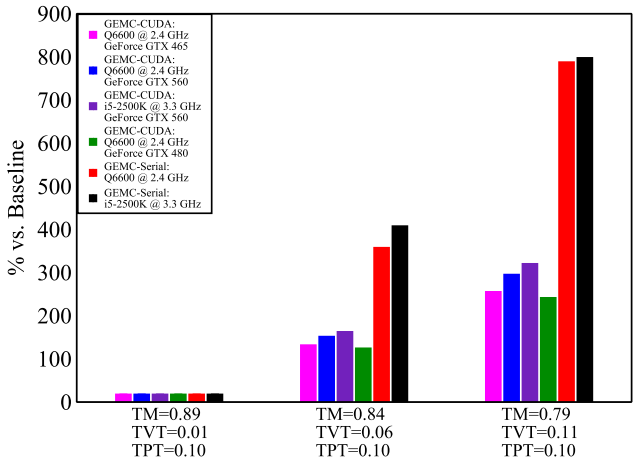

Fig. 8. Performance of Gibbs ensemble Monte Carlo simulations for varying ratios of move types. Data are presented as a percent difference relative to the baseline case of 1% volume transfer (TVT), 89% particle displacement (TM) and 10% particle transfer (TPT). Data were calculated for a system containing 2048 particles with a Lennard-Jones cut-off of 3.0σ.

Bridge architecture i5-2500K CPU operating at 3.3 GHz was approximately 50% faster than the 2.4 GHz the *Kentsfield* Q6600 chip, with the majority of this difference resulting from the 37% increase in CPU clock speed of the i5-2500K compared to the Q6600. The GeForce GTX 480 was found to be fastest of the GPUs tested, despite having approximately 20% slower CUDA core and 10% slower memory clock rates than the GeForce 560. This is because the GeForce GTX 480 has 480 CUDA cores compared to 384 for the GTX 560 Ti. The choice of CPU was found to have a negligible impact the performance of the GPU-accelerated code, despite the fact that some portions of the code (reading configuration parameters, output functionality) were contained on the host (CPU) side.

For the Gibbs ensemble simulations, the addition of the particle swap and volume moves raises computational demands on the GPU, but provides greater opportunity for optimization. In this case, the ratio of moves was set to 1% volume exchange, 10% particle swap and 89% particle displacement; ratios that are typical of production GEMC simulations. Timing data are presented for system sizes between 512 and 131,072 particles in Fig. 7. Numerical data are presented in Table 2. For the largest system studied ($N = 131{,}072$) the GPU GEMC simulations were 29.1 times faster than the serial, CPU-bound code. For the Gibbs ensemble, the breakeven point was found at 600 particles; for smaller system sizes the CPU code was faster than the GPU code. This was due to the extra overhead of memory communications for the second box, and poor performance of the parallel volume move for small system sizes.

The performance of GEMC simulations may be affected significantly by the selected distribution of moves. There was a negligible difference in the computational cost of the particle displacement and particle swap moves on the GPU. The volume exchange move is the most computationally demanding and increasing the frequency of volume exchange moves carries a substantial performance penalty. In Fig. 8 timing data for simulations performed with 1%–11% volume swap moves are presented. These data are provided as percentage change from the baseline case (1% volume swap, 10% particle swap, 89% particle displacement). The data show that increasing the number of volume moves, while reducing the number of particle displacements, led to large changes in the run time of both the serial and GPU codes. In the GPU-GEMC code, increasing the fraction of volume swaps performed from 1% to 11% increased the simulation run time by a factor of 2.3. For comparison, using the same distribution of moves in the serial code produced a result that was 7.9 times slower than the original case. Consistent with prior calculations, these data show the GPU GEMC code exhibits significantly better scaling with increasing workload compared to the CPU GEMC implementation.

5. Conclusions

In this work, an implementation of Gibbs ensemble Monte Carlo on the GPU has been demonstrated. Parallelization of routines to calculate pair-wise energies was performed using the method of energetic decomposition, where each pair interaction was calculated by a particular thread. This method is naturally well suited to the architecture of the GPU, but has limitations. For small system sizes, $N < 600$ particles, it was found that overhead from memory transfers between system RAM and the GPU led to worse performance than a traditional serial CPU bound code. However, for larger systems, the GPU code was found to be significantly faster; for the largest system studied ($N = 131{,}072$), the GPU accelerated Gibbs ensemble code was 29.1 times faster than the serial code. In other words, for the 131,072 particle system, a single GPU offered performance equivalent to 29 CPU Intel i5-2500K cores running in embarrassingly parallel mode. For system sizes between 512 and 4096 particles, the computational effort was independent of system size, suggesting the GPU had significant unused computational capacity that could be used, in principle, for calculations related to advanced sampling techniques, such as configurational-bias, with little or no penalty in run time. It should be noted that the performance data presented in this work were determined using double precision arithmetic for all calculations performed on the GPU. Preliminary tests by our group have shown the GPU code was approximately twice as fast running single precision mode compared to double precision mode with a negligible loss in accuracy, therefore the presented performance data represent a conservative estimate of the peak performance of the GPU code. For Lennard-Jones beads, or small molecules, the hybrid methodology (a combination of energetic decomposition and embarrassingly parallel methods) used by Kim et al. in canonical Monte Carlo simulations is more efficient [53], however, for the molecules and system sizes typical in biomolecular simulations, the methodology described in this work is expected to provide the best overall performance.

Acknowledgments

Financial support from the National Science Foundation (NSF CBET-0730786, OCI-1148168) and the Wayne State University Research Enhancement Program is gratefully acknowledged.

Appendix. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.cpc.2013.06.020.

References

[1] A.Z. Panagiotopoulos, Mol. Phys. 61 (1987) 813.
[2] B. Smit, P. Desmedt, D. Frenkel, Mol. Phys. 68 (1989) 931.
[3] J.J. de Pablo, M. Laso, J.I. Siepmann, U.W. Suter, Mol. Phys. 80 (1993) 55.
[4] M.G. Martin, J.I. Siepmann, J. Phys. Chem. B 103 (1999) 4508.
[5] J.I. Siepmann, D. Frenkel, Mol. Phys. 75 (1992) 59.
[6] T.J.H. Vlugt, M.G. Martin, B. Smit, J.I. Siepmann, R. Krishna, Mol. Phys. 94 (1998) 727.
[7] C.D. Wick, J.I. Siepmann, Macromolecules 33 (2000) 7207.
[8] D.A. Kofke, Mol. Phys. 78 (1993) 1331.
[9] A.M. Ferrenberg, R.H. Swendsen, Phys. Rev. Lett. 61 (1988) 2635.
[10] J.J. Potoff, J.R. Errington, A.Z. Panagiotopoulos, Mol. Phys. 97 (1999) 1073.
[11] A.M. Ferrenberg, D.P. Landau, P. Peczak, J. Appl. Phys. 69 (1991) 6153.
[12] A.M. Ferrenberg, R.H. Swendsen, Phys. Rev. Lett. 63 (1989) 1195.
[13] A.Z. Panagiotopoulos, Mol. Phys. 62 (1987) 701.
[14] S.C. McGrother, K.E. Gubbins, Mol. Phys. 97 (1999) 955.
[15] X. Peng, J.S. Zhao, D.P. Cao, J. Colloid Interface Sci. 310 (2007) 391.
[16] C. Lastoskie, K.E. Gubbins, N. Quirke, Langmuir 9 (1993) 2693.
[17] T. Okayama, J. Yoneya, T. Nitta, Fluid Phase Equilib. 104 (1995) 305.
[18] J.W. Jiang, S.I. Sandler, Langmuir 19 (2003) 5936.
[19] M.B. Sweatman, N. Quirke, Mol. Simul. 30 (2004) 23.
[20] B. Chen, J.I. Siepmann, M.L. Klein, J. Phys. Chem. B 105 (2001) 9840.
[21] X.S. Zhao, B. Chen, S. Karaborni, J.I. Siepmann, J. Phys. Chem. B 109 (2005) 5368.
[22] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kale, K. Schulten, J. Comput. Chem. 26 (2005) 1781.
[23] S. Plimpton, J. Comput. Phys. 117 (1995) 1.
[24] D.A. Case, T.E. Cheatham, T. Darden, H. Gohlke, R. Luo, K.M. Merz, A. Onufriev, C. Simmerling, B. Wang, R.J. Woods, J. Comput. Chem. 26 (2005) 1668.
[25] J.A. Anderson, C.D. Lorenz, A. Travesset, J. Comput. Phys. 227 (2008) 5342.
[26] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, J. Chem. Theory Comput. 4 (2008) 435.
[27] S.J. Zara, D. Nicholson, Mol. Simul. 5 (1990) 245.
[28] D.E. Ulberg, K.E. Gubbins, Mol. Simul. 13 (1994) 205.
[29] D.M. Jones, J.M. Goodfellow, J. Comput. Chem. 14 (1993) 127.
[30] G.S. Heffelfinger, Comput. Phys. Comm. 128 (2000) 219.
[31] G.S. Heffelfinger, M.E. Lewitt, J. Comput. Chem. 17 (1996) 250.
[32] R.C. Ren, G. Orkoulas, J. Chem. Phys. 126 (2007).
[33] C.J. O'Keeffe, G. Orkoulas, J. Chem. Phys. 130 (2009).
[34] C.J. O'Keeffe, R.C. Ren, G. Orkoulas, J. Chem. Phys. 127 (2007).
[35] B. Mehlig, D.W. Heermann, B.M. Forrest, Phys. Rev. B 45 (1992) 679.
[36] L.D.J.C. Loyens, B. Smit, K. Esselink, Mol. Phys. 86 (1995) 171.
[37] F.A. Brotz, J.J. Depablo, Chem. Eng. Sci. 49 (1994) 3015.
[38] A. Chen, C.S. Hirtzel, Sep. Technol. 4 (1994) 167.
[39] A. Chen, C.S. Hirtzel, Mol. Phys. 82 (1994) 263.
[40] A. Chen, C.S. Hirtzel, Int. J. Supercomput. Appl. High Perform. Comput. 8 (1994) 54.
[41] G.S. Heffelfinger, D.M. Ford, Mol. Phys. 94 (1998) 659.
[42] D.M. Ford, G.S. Heffelfinger, Mol. Phys. 94 (1998) 673.
[43] K. Esselink, L.D.J.C. Loyens, B. Smit, Phys. Rev. E 51 (1995) 1560.
[44] L. Durant, T. Szalay, R. McClellan, Hist. GPU (2011).
[45] T.D. Nguyen, C.L. Phillips, J.A. Anderson, S.C. Glotzer, Comput. Phys. Comm. 182 (2011) 2307.
[46] C.L. Phillips, J.A. Anderson, S.C. Glotzer, J. Comput. Phys. 230 (2011) 7191.
[47] W.M. Brown, P. Wang, S.J. Plimpton, A.N. Tharrington, Comput. Phys. Comm. 182 (2011) 898.
[48] J.C. Phillips, J.E. Stone, K. Schulten, Adapting a message-driven parallel application to GPU-accelerated clusters, in: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, IEEE Press, Austin, Texas, 2008, p. 1.
[49] A.W. Gotz, M.J. Williamson, D. Xu, D. Poole, S. Le Grand, R.C. Walker, J. Chem. Theory Comput. 8 (2012) 1542.
[50] T. Preis, P. Virnau, W. Paul, J.J. Schneider, J. Comput. Phys. 228 (2009) 4468.
[51] B. Block, P. Virnau, T. Preis, Comput. Phys. Comm. 181 (2010) 1549.
[52] A. Frezzotti, G.P. Ghiroldi, L. Gibelli, AIP Conf. Proc. 1333 (2011) 884.
[53] J. Kim, J.M. Rodgers, M. Athenes, B. Smit, J. Chem. Theory Comput. 7 (2011) 3208.
[54] J. Kim, B. Smit, J. Chem. Theory Comput. 8 (2012) 2336.
[55] J. Kim, L.C. Lin, R.L. Martin, J.A. Swisher, M. Haranczyk, B. Smit, Langmuir 28 (2012) 11923.
[56] J. Kim, R.L. Martin, O. Rubel, M. Haranczyk, B. Smit, J. Chem. Theory Comput. 8 (2012) 1684.
[57] N.B. Wilding, Phys. Rev. E 52 (1995) 602.
[58] J.J. Potoff, A.Z. Panagiotopoulos, J. Chem. Phys. 112 (2000) 6411.
[59] W. Shi, J.K. Johnson, Fluid Phase Equilib. 187 (2001) 171.
[60] http://www.cstl.nist.gov/srs/LJ_PURE/index.htm.
[61] J.S. Rowlinson, R.L. Swinton, Liquids and Liquid Mixtures, third ed., Butterworths, London, 1982.
[62] V.G. Privman, G.L. Trigg, Encyclopedia of Applied Physics, Wiley VCH, Berlin, 1998.
[63] J.J. Potoff, A.Z. Panagiotopoulos, J. Chem. Phys. 109 (1998) 10914.

Please cite this article in press as: J. Mick, et al., GPU-accelerated Gibbs ensemble Monte Carlo simulations of Lennard-Jonesium, Computer Physics Communications (2013), http://dx.doi.org/10.1016/j.cpc.2013.06.020