JCTC Journal of Chemical Theory and Computation

ACEMD: Accelerating Biomolecular Dynamics in the Microsecond Time Scale

M. J. Harvey,*,† G. Giupponi,‡ and G. De Fabritiis§

Information and Communications Technologies, Imperial College London, South Kensington, London, SW7 2AZ, United Kingdom, Department de Fisica Fundamental, Universitat de Barcelona, Carrer Marti i Franques 1, 08028 Barcelona, Spain, and Computational Biochemistry and Biophysics Lab (GRIB-IMIM), Universitat Pompeu Fabra, Barcelona Biomedical Research Park (PRBB), C/ Doctor Aiguader 88, 08003 Barcelona, Spain

Received February 4, 2009

Abstract: The high arithmetic performance and intrinsic parallelism of recent graphical processing units (GPUs) can offer a technological edge for molecular dynamics simulations. ACEMD is a production-class biomolecular dynamics (MD) engine supporting CHARMM and AMBER force fields. Designed specifically for GPUs it is able to achieve supercomputing scale performance of 40 ns/day for all-atom protein systems with over 23 000 atoms. We provide a validation and performance evaluation of the code and run a microsecond-long trajectory for an all-atom molecular system in explicit TIP3P water on a single workstation computer equipped with just 3 GPUs. We believe that microsecond time scale molecular dynamics on cost-effective hardware will have important methodological and scientific implications.

I. Introduction

The simulation of mesoscopic scales (microseconds to milliseconds) of macromolecules continues to pose a challenge to modern computational biophysics. While the fundamental thermodynamic framework behind the simulation of macromolecules is well characterized, exploration of biological time scales remains beyond the computational capacity routinely available to many researchers. This has significantly inhibited the widespread use of molecular simulations for in silico modeling and prediction.[1]

Recently, there has been a renewed interest in the development of molecular dynamics simulation techniques. D. E. Shaw Research[2] has fostered several significant algorithmic improvements including midpoint[3] and neutral-territory methods[4] for the summation of nonbonded force calculations, a new molecular dynamics package called Desmond[5] and Anton,[2] a parallel machine for molecular dynamics simulations that uses specially designed hardware. Other parallel MD codes, such as Blue matter,[6] NAMD,[7] and Gromacs4,[8] have been designed to perform parallel MD simulations across multiple independent processors, but latency and bandwidth limitations in the interconnection network between processors reduces parallel scaling unless the size of the simulated system is increased with processor count. Furthermore, dedicated, highly parallel machines are usually expensive and not reservable for long periods of time due to cost constraints and allocation restrictions.

A further line of development of MD codes consists of using commodity high-performance accelerated processors.[1] This approach has become an active area of investigation, particularly in relation to the Sony−Toshiba−IBM Cell processor[9] and graphical processing units (GPUs). Recently, De Fabritiis[9] implemented an all-atom biomolecular simulation code, CellMD, targeted to the architecture of the Cell processor (contained within the Sony Playstation3) that reached a sustained performance of 30 Gflops with a speed up of 19 times compared to the single CPU version of the code. At the same time, a port of the Gromacs code for implicit solvent models[10] was developed and used by the Folding@home distributed computing project[11] on a distrib-

* To whom correspondence should be addressed. E-mail: m.j.harvey@imperial.ac.uk, gianni.defabritiis@upf.edu.

† Imperial College London.

‡ Universitat de Barcelona.

§ Universitat Pompeu Fabra.

uted network of Playstation3s. Similarly, CellMD was used in the PS3GRID.net project[12] based on the BOINC platform[13] moving all-atom MD applications into a distributed computing infrastructure.

Pioneers in the use of GPUs for production molecular dynamics[11] had several limitations imposed by the restrictive, graphics-orientated OpenGL programming model[14] then available. In recent years, commodity GPUs have acquired nongraphical, general-purpose programmability and undergone a doubling of computational power every 12 months, compared to 18−24 months for traditional CPUs.[1] Of the devices currently available on the market, those produced by Nvidia offer the most mature programming environment, the so-called compute unified device architecture (CUDA),[15] and have been the focus of the majority of investigation in the computational science field.

Several groups have lately shown results for MD codes which utilize CUDA-capable GPUs. Stone et al.[16] demonstrated the GPU-accelerated computation of the electrostatic and van der Waals forces, reporting a 5.4 times speed up with respect to a conventional CPU. Meel et al.[17] described an implementation for simpler Lennard−Jones atoms which achieved a net speed up of up to 40 times over a conventional CPU. Unlike the former, the whole simulation is performed on the GPU. Recently, Phillips et al. reported experimental GPU acceleration of NAMD[18] yielding speed ups of up to 7 times over NAMD 2.6. Pande et al. announced OpenMM,[19] a library of GPU kernels for MD, derived from their work on Folding@Home.[11] Though very fast, the OpenMM kernels use direct summation of nonbonded terms, making its use suitable mainly for implicit solvent systems.

More widely in the field of computational chemistry, investigation has proceeded into GPU acceleration[20] of a variety of quantum chemical methods, including quantum Monte Carlo,[21] correlation,[22] and self-consistent field[23] methods.

In this work, we report on a molecular dynamics program called ACEMD which is optimized to run on Nvidia GPUs and which has been developed with the aim of advancing the frontier of molecular simulation toward the ability to routinely perform *microsecond-scale* simulations. ACEMD maximizes performance by running the whole computation on the GPU rather than offloading only selected computationally expensive parts. We developed ACEMD to implement all features of a typical MD simulation including those usually required for production simulations such as particle−mesh Ewald (PME)[24] calculation of long-range electrostatics, thermostatic control, and bond constraints. The default force-field format used by ACEMD is CHARMM[25] and Amber[26] force fields. ACEMD also provides a scripting interface to control and program the molecular dynamics run to perform complex protocols like umbrella sampling, steered molecular dynamics and sheared boundary conditions, and metadynamics simulations.[27]

II. GPU Architecture

The G80 and subsequent G200 generations of Nvidia GPU architectures are designed for data-parallel computation, in which the same program code is executed in parallel on many

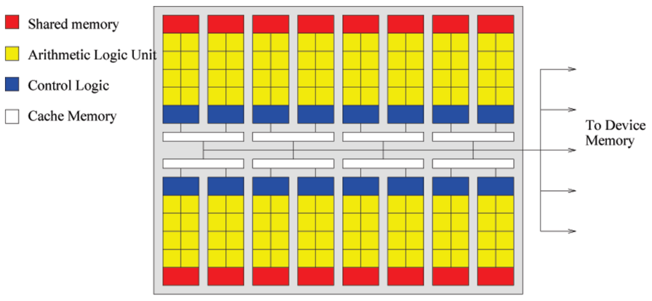

Figure 1. Nvidia GPU design is based around an 8-core single-program, multiple data (SPMD) processor. Each core has local storage provided by the register file and access to a shared memory region. Read/write access to the main device memory is uncached, except for some specific read-only access modes. The above figure represents the G80-series device with 16 such processors, while the contemporary G200 contains 30 (Tesla C1060, GTX280), giving 240 cores.

Table 1. Summary of Characteristics of First- and Second-Generation Nvidia GPU Compute Devices, with a Contemporary Intel Xeon Shown for Comparison[a]

	Tesla C870 (G80)	Tesla C1060 (G200)	Intel Xeon 5492
cores	128	240	4
clock (GHz)	1.350	1.296	3.4
mem bandwidth (MB/s)	77	102	21
Gflops (sp/dp)	512/−	933/78	108/54
power (W)	171	200	150
year	2007	2008	2008

[a] Data taken from manufacturers' data sheets. (sp stands for single precision and dp for double precision).

data elements. The CUDA programming model, an extended C-like language for GPUs, abstracts the implementation details of the GPU so that the programmer may easily write code that is portable between current and future GPUs. Nvidia GPU devices are implemented as a set of multiprocessor (MP) devices, each of which is capable of synchronously executing 32 program threads in parallel (called a warp) and managing up to 1024 concurrently (Figure 1).[55] Current Nvidia products based on these devices are able to achieve up to 933 Gflops in single precision. By comparison, a contemporary quad-core Intel Xeon CPU is capable of approximately 54 Gflops double precision/108 Gflops single precision.[28,29] A brief comparison of the characteristics of these devices is given in Table 1. Each MP has a set of 32-bit registers, which are allocated as required to individual threads, and a region of low-latency shared memory that is accessible to all threads running on it. The MP is able to perform random read/write access to external memory. Access to this global memory is uncached and so incurs the full cost of the memory latency (up to 400 cycles). However, when accessed via the GPU's texturing units, reads from arrays in global memory are cached, mitigating the impact of global memory access for certain read patterns. Furthermore, the texture units are capable of performing linear interpolation of values into multidimensional (up to 3D) arrays of floating point data.

While the older G80 architecture supported only single-precision IEEE-754 floating point arithmetic, the newer G200 design also supports double-precision arithmetic, albeit at a much lower relative speed. The MP has special hardware support for reciprocal square root, exponentiation, and trigonometric functions, allowing these to be computed with low latency but at the expense of slightly reduced precision.

Program fragments written to be executed on the GPU are known as *kernels* and executed in *blocks*. Each block consists of multiple instances of the kernel, called *threads*, which are run concurrently on a single multiprocessor. The number of threads in a block is limited by the resources available on the MP, but multiple blocks may be grouped together as a *grid*. The CUDA runtime, in conjunction with the GPU hardware itself, is responsible for efficiently scheduling the execution of a grid of blocks on available GPU hardware. CUDA does not presently provide a mechanism for transparently using multiple GPU devices for parallel computation. For full details of the CUDA environment, the reader is referred to the SDK documentation.[30]

III. Molecular Dynamics on the GPU

ACEMD implements all features of an MD simulation on a CUDA-compatible GPU device, including those usually required for production simulations in the NVT ensemble (i.e., bonded and nonbonded force term computation, velocity-Verlet integration, Langevin thermostatic control, smooth Ewald long-range electrostatics (PME),[24,31] and hydrogen bond constraints). Also implemented is the hydrogen mass repartitioning scheme described in ref 32 and used, for instance, in codes such as Gromacs, which allows an increased time step of up to 4 fs. The code does not presently contain a barostat, so simulations in the NPT ensemble are not possible. However, it is noted that with large molecular systems, changes in volume due to the pressure control are very limited after an initial equilibration making NVT simulations viable for production runs. ACEMD supports the CHARMM27 and Amber force fields, PDB, PSF, and DCD file formats,[33] check pointing, and input files compatible with widely used MD codes. It is extensible via a C-based plugin interface and TCL scripting (see ACEMD online manual for details[34]).

The computation of the nonbonded force terms dominates the computational cost of MD simulation, and it is therefore important to use an efficient algorithm. As in refs 9 and 17, we implement a cell-list scheme in which particles are binned according to their coordinates. On all-atom biomolecular systems a cutoff of $R = 12$ Å with bins $R/2$ gives an average cell population of approximately 22 atoms. This is comparable to the warp size of 32 for current Nvidia GPUs. In practice, however, transient density fluctuations can lead to the cell population exceeding the warp size. Consequently, the default behavior of ACEMD is to assume a maximum cell population of 64. The code may also accommodate a bin size of R for coarse-grained simulations. The cell-list construction kernel processes one particle per thread, with each thread computing the cell in which its atom resides. To permit concurrent manipulation of a cell-list array, atomic memory operations are used.

The nonbonded force computation kernel processes a single cell per thread block, computing the full Lennard−Jones

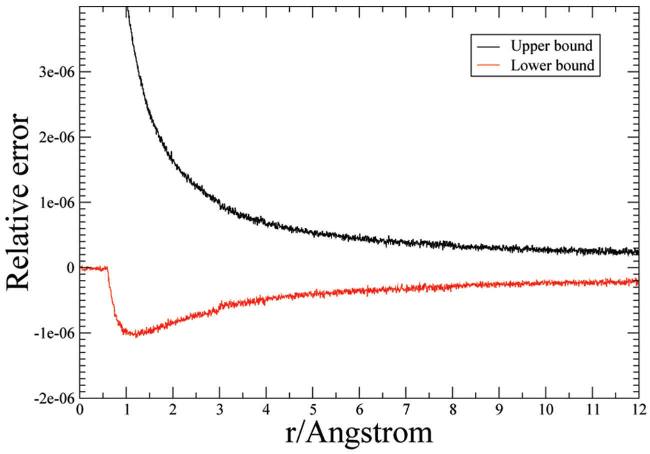

Figure 2. Bounds (running average) of the relative error $(E_{\mathrm{interp}} - E_{\mathrm{calcd}})/(E_{\mathrm{calcd}})$ between directly calculated and lookup table (linear interpolation, $n = 4096$) values of the van der Waals potential. Error has a period of $(R_{\max})/(n)$.

and electrostatic force on each particle residing within it. All of the cells within R of the current cell (including a copy of the cell itself) are loaded into shared memory in turn. Each thread then computes the force on its particle by iterating over the array in shared memory. In contrast to CPU implementations, reciprocal forces are not stored for future use (i.e., the force term F_{ij} is not saved for reuse as F_{ji}), because of the relatively high cost of global memory access.[56] The texture units are used to assist the calculation of the electrostatic and van der Waals terms by providing linearly interpolated values for the radial components of those functions from lookup tables. The interpolation error is low and does not affect the energy conservation properties of NVE simulations (Figure 2).

In production runs, the relative force error compared to a reference simulation performed in double precision is consistently less than 10^{-4}, below the 10^{-3} error considered the maximum acceptable for biomolecular simulations.[5] Particle-mesh Ewald (PME)[31] evaluation of long-range electrostatics is also supported by a dedicated kernel. All parts of this computation are performed on the GPU, with support from the Nvidia FFT library.[35] For PME calculations, a cutoff of $R = 9.0$ Å accepted as provided sufficient accuracy, permitting the maximum cell population to be limited to 32 atoms.

To support CHARMM and AMBER force fields, it is necessary to selectively exclude or scale nonbonded force terms between atoms that share an explicit bond term. The indices of excluded and 1−4 scaled pairs are stored in bitmaps, allowing any pair of particles with indices i,j such that $|i - j| \leq 64$ to be excluded or scaled.[16] Exclusions with larger index separations are also supported in order to accommodate, for example, disulfide bonds, but the additional book-keeping imposes a minor reduction in performance. Because the atoms participating in bonded terms are spatially localized, it is necessary only to make exclusion tests for interactions between adjacent cells despite, for cells of $R/2$, the interaction halo being two cells thick. A consequent optimization is the splitting of the nonbonded

Table 2. Energy Change in the NVE Ensemble per Nanosecond per Degree of Freedom (dof) in K_bT Units for Dihydrofolate Reductase (DHFR) Using Different Integration Time Steps, Constraints, and Hydrogen Mass Repartitioning (HMR) Schemes

time step (fs)	constraints	HMR	K_bT/ns/dof
1	no	no	0.00021
2	yes	no	−0.00082
4	yes	yes	−0.00026

force kernel into two versions, termed *inner* and *outer*, which, respectively, include and omit the test.

Holonomic bond constraints are implemented using the M-shake algorithm[36] and RATTLE for velocity constraints[37] within the velocity Verlet integration scheme.[38] M-shake is an iterative algorithm, and in order to achieve acceptable convergence it is necessary to use double-precision arithmetic (a capability available only on G200/architecture 1.3 class devices). For the pseudorandom number source for the Langevin thermostat we use a Mersenne twister kernel, modified from the example provided in the CUDA SDK.

IV. Single-Precision Floating-Point Arithmetic Validation

ACEMD uses single-floating point arithmetic because the performance of GPUs on single precision is much higher than double precision and the limitation of a single floating point can be controlled well for molecular dynamics.[8,9] Nevertheless, we validate in this section the conservation properties of energy in a NVT simulation using rigid and harmonic bonds, as constraints have shown to be more sensitive to numerical precision. Potential energies were checked against NAMD values for the initial configuration of a set of systems, including disulfide bonds, ionic system, protein, and membranes, in order to verify the correctness of the force calculations by assuring that energies were identical within 6 significant figures. The Langevin thermostat algorithm was tested for three different damping frequencies $\gamma = 0.1$, 0.5, and 1.0 with a reference temperature of $T = 300$ K and both with and without constraints.

The test simulations consist of nanosecond runs of dihydrofolate reductase (DHFR) joint AMBER-CHARMM benchmark with volume 62.233 × 62.233 × 62.233 Å^3 (a total of 23 558 atoms).[5] Each simulation system was first equilibrated at a temperature of $T = 300$ K and then relaxed in the NVE ensemble. A reference simulation with harmonic bonds and time step $dt = 1$ fs was also performed, as well as simulations with $dt = 2$ fs using rigid constraints and with $dt = 4$ fs with rigid constraints and hydrogen mass repartitioning (HMR) with a factor 4.0, as in ref 39. In Table 2 we show the energy change per nanosecond per degree of freedom in units of K_bT, which is similar to other single- and double-precision codes MD.[40] We note that even when using bigger time steps and a combination of M-shake and hydrogen mass repartitioning, energy conservation is reasonably good and much slower than the time scale at which the thermostat would act. Hydrogen mass repartitioning is an elegant way to increase the time step up to 4 fs by increasing the momentum of inertia of groups of atoms bonded to

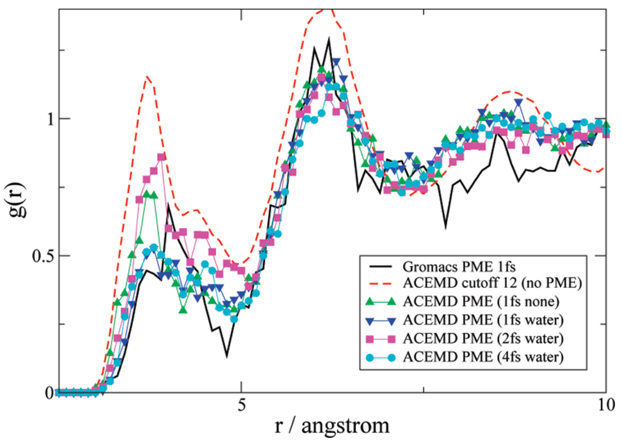

Figure 3. Plot of Na−Na pair distribution functions for a 1 M NaCl water box as in ref 41.

hydrogen atoms. The mass of the bonded heavy atoms to hydrogens is repartitioned among hydrogen atoms, leaving the total mass of the system unchanged. As individual atom masses do not appear in the expression for the equilibrium distribution, this repartition affects only the dynamic properties of the system not the equilibrium distribution. Following ref 32, a factor 4 for hydrogens affects only marginally the diffusion and viscosity of TIP3P water (which is in any case inaccurate when compared to experimental data). A similar speed up could also be obtained by using a smaller time step with the evaluation of the long-range electrostatic terms every other time step.

We also validated the implementation of the PME algorithm to compute long-range electrostatics forces. We ran a set of simulations using different time steps and algorithms as above ($dt = 1$ and 2 fs rigid bonds, $dt = 4$ fs rigid bonds and hydrogen mass repartitioning) on a 40.5 × 40.5 × 40.5 Å^3 box of 1 M solution of NaCl in water (6461 atoms), as in ref 41. PME calculations were performed with a 64 × 64 × 64 grid size. Two simulations of the same system were used as reference: one with Gromacs[8] with PME as in ref 41 and the other using ACEMD with an electrostatic cutoff of 12 Å without PME. We calculated the Na−Na pair distribution function $g(r)$ in Figure 3 in order to compare the simulation results for different simulations and methods, as from ref 41 Na−Na $g(r)$ results as the quantity more sensitive to different methods for electrostatics calculations. We note that for all integration time steps used, ACEMD agrees well with the reference simulation made with Gromacs. In addition, using PME gives consistently better results than using a 12 Å cutoff for this simple homogeneous system, as expected. A direct validation of the pair distribution function with the hydrogen mass repartitioning method is also shown in Figure 3, comparing the $g(r)$ for time step = 1, 2, and 4.

V. Performance

The current implementation of ACEMD is parallelized in a task parallel manner designed to scale across just 3 GPUs attached to a single host system. A simple force−decomposition scheme[42] is used, in which each GPU computes a subset of the force terms. These force terms are summed

Table 3. Performance of ACEMD on the DHFR Benchmark[a]

Program	CPU Cores and GPUs	ms/step
ACEMD	1 CPU, 1 GPU (240 cores)	17.55
ACEMD	3 CPU, 3 GPU (720 cores)	7.56
NAMD2.6	128 CPU (64 nodes)	9.7
NAMD2.6	256 CPU (128 nodes)	7.0
Desmond	32 CPU (16 nodes)	11.5
Desmond	64 CPU (32 nodes)	6.3
Gromacs4	20 CPU (5 nodes)	7

[a] GPUs are Nvidia GTX 280. NAMD, Desmond and Gromacs performances are indicative of the orders of magnitude speed up obtained with GPUs and ACEMD as they are all performed on different CPU systems (from refs 5 and 8 Gromacs figures interpolated from Figure 6 of ref 8). Desmond and Gromacs use SSE vector instructions and single precision floating-point numbers.

by the host processor and the total force matrix transferred back to each GPU which then performs integration of the whole system. ACEMD dynamically load balances the computation across the GPUs. This allows the simulation of heterogeneous molecular systems and also accommodates variation due to host system architecture (for example, different speed GPUs or GPU-host links). For simulations requiring PME, a heterogeneous task decomposition is used, with a subset of GPUs dedicated to PME computation.

The performance benchmark is based on the DHFR molecular system with a cutoff of $R = 9$ Å, switched at 7.5 Å, $dt = 4$ fs, PME for long-range electrostatic with $64 \times 64 \times 64$ grid size, and fourth-order interpolation, M-shake constraints for hydrogen bonds and hydrogen mass repartitioning. All simulations were run on a PC equipped with 4 Nvidia GPU GTX 280 cards at 1.3 GHz (just 3 GPUs used for these tests), a quad-core AMD Phenom processor (2.6 GHz), MSI board with AMD790 FX chipset, 4GB RAM running Fedora Core 9, CUDA toolkit 2.0, and the Nvidia graphics driver 177.73. Performance results reported in Table 3 indicate that ACEMD requires 17.55 ms per step with the DHFR system and 7.56 ms per step when run in parallel over the 3 GPUs. As expected by the simple task decomposition scheme, ACEMD achieves a parallel efficiency of 2.3 over 3 GPUs. Further device-to-device communication directives may substantially improve these results as they will enable the use of spatial-decomposition parallelization strategies, such as neutral territory (NT) schemes.[4] Comparing directly the maximum performance of ACEMD on the DHFR system with results of various MD programs from ref 5 we obtain a performance approaching that of 256 CPU cores using NAMD and 64 using Desmond on a cluster with fast interconnect. Using hydrogen mass repartitioning and a time step of 4 fs integration time step it is possible to simulate trajectories of over 45 ns per day with 3 GPUs and almost 20 ns per day with a single GPU. A highly optimized code such as Gromacs4 requires only 20 CPU cores to deliver similar performance to 3 GPUs on DHFR,[8] but the calculations are not identical as, for instance, there are several optimizations applied to water.

Representative performance data for ACEMD and the GPU-accelerated version of NAMD[18] for the apoA1 benchmark system (92 224 atoms) is given in Table 4. Differences between the simulation and hardware configurations prevent a direct comparison, but it is salient to note that because the enhanced NAMD retains the spatial decomposition parallelism it is able to scale across multiple GPU-equipped hosts, while ACEMD is designed for optimal performance on a small number of GPUs.

Table 4. Performance of ACEMD and NAMD on the apoA1 Benchmark[a]

program	CPU cores and GPUs	ms/step
ACEMD	1 CPU, 1 GPU (1 node)	73.4
ACEMD	3 CPU, 3 GPU (1 node)	32.5
NAMD	4 CPU, 4 GPU (1 node)	87
NAMD	16 CPU, 16 GPU (4 nodes)	27
NAMD	60 CPU (15 nodes)	44

[a] ACEMD run using Nvidia GTX 280 GPUs ($R = 9$ Å, PME every step), NAMD ($R = 12$ Å, PME every 4 steps) run with G80-series GPUs (approximately half as fast) NAMD performance data taken from ref 18.

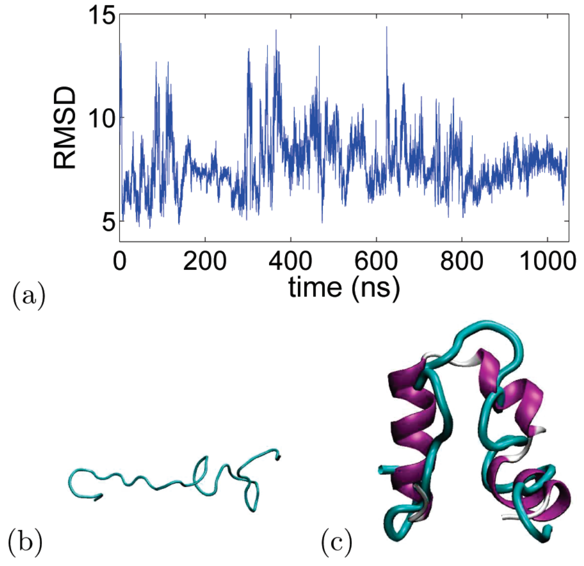

Figure 4. (a) rmsd of the backbone of the protein during the microsecond simulation starting from the unfolded configuration (b) of the Villin system with 13 701 atoms (TIP3P water not shown for clarity). Within our simulation window the minimum rmsd was 4.87 Å for which the resulting best structure is overlapped with the crystal structure in c.

VI. Microsecond Simulations on Workstation Hardware

To provide a direct demonstration that molecular simulations have now entered the microsecond regime *routinely* we perform a microsecond long trajectory performed on a workstation-class PC. We use for this task the chicken Villin headpiece (HP-35) structure, one of the smallest polypeptides with a stable globular structure comprising three alpha helices placed in a “U”-shaped form, as shown in Figure 4c. Due to its small size, it is commonly used as a subject in long molecular simulations for folding studies, for instance,ref 43, which uses highly parallel distributed computing to compute many trajectories to fully sample the phase space of the folding process. Additionally, mutanogesis studies on fold-

Table 5. Performance of ACEMD on the DHFR, apoA1 Benchmark and Villin Test on 1 and 3 GPUs up to 720 Cores[a]

system	CPUs and GPUs	ns/day
DHFR	1 CPU, 1 GPU (240 cores)	19.7
DHFR	1 CPU, 3 GPUs (720 cores)	45.7
apoA1	1 CPU, 1 GPU (240 cores)	4.6
apoA1	3 CPU, 3 GPUs (720 cores)	10.6
Villin	3 CPU, 3 GPUs (720 cores)	66.0

[a] ACEMD run using Nvidia GTX 280 GPUs on real production runs (R = 9 Å, PME every step, time step 4 fs, constraints, and Langevin thermostat).

ing[44] have been performed using biased methods to accelerate the sampling.

The Villin headpiece (PDB:1YRF) was fully solvated in TIP3P water and Na−Cl at 150 mM (a total of 13 701 atoms) using the program VMD[45] and the CHARMM force field. The system was then equilibrated at 300 K and 1 atm for 10 ns using NAMD2.6[7] with a cutoff of 9 Å, PME with a 48 × 48 × 48 grid, constraints for all H bond terms, and a time step of 2 fs. Simulations with ACEMD were performed using an NVT ensemble, hydrogen mass repartitioning, and a time step of 4 fs. Starting from the final equilibrium configuration of NAMD, we run ACEMD at 450 K for 40 ns until the system was completely unfolded (movie available in ref 46). The resulting extended configuration (Figure 4b) was then used as the starting point of a microsecond long single trajectory at a temperature of 305 K. Figure 4a shows the rmsd of the backbone of the protein along the trajectory. The minimum rmsd was 4.87. The protein seems to sample quite often the overall shape of the crystal structure yet not converged toward it (Figure 4c). As this structure is expected to fold in 4−5 μs, we plan to extend the dynamics in the future along with any newer and faster version of ACEMD (for instance, using the new Nvidia GTX295 cards or, more likely, quad GPU Tesla S1075 units). An important consideration with regard to the force field is the following: with molecular simulations approaching microseconds, it is clear that the accuracy of the force fields will become more and more important. In particular, this system has been shown to be very sensitive to the force field used[44] (CHARMM seems to converge poorly toward the folded structure).

The production run on a PC equipped with ACEMD and 3 Nvidia GPUs (720 cores) required approximately 15 days (66 ns/day) (see Table 5) and probably represents the limit for current hardware and software implementation, while 5 μs should be obtainable in the near future using a 4-way GTX295-based system with 8 GPU cores. Using currently available commodity technology, the construction of computer systems with up to 8 directly attached GPUs has been demonstrated.[16,47] GPUs attach to the host system using the industry-standard PCI-Express interface.[48] This interface is characterized by a bandwidth comparable to that of main system memory (up to 8 GB/s for 16 lane PCIe 2.0 links typically used by graphics cards) but with a relatively higher latency.

The GPU resource requirements of the nonbonded kernel make it possible for up to 8 independent blocks to be processed simultaneously per multiprocessor. The limit of parallelization for the execution of the nonbonded kernel occurs when all blocks may be processed simultaneously by the available multiprocessors. Thus, for instance, a cubic simulation box with l = 66 Å and cell size 6 Å would scale over 167 multiprocessors (1336 cores) or 6 G200-class GPUs. Figure 5 shows the runtime of the inner and outer nonbonded kernels on a water box as a function of block count per kernel invocation. The minimum computation time for the fully parallel case would be 3.4 ms/step on current hardware. To further improve performance, optimization of the kernel or further subdivision of the computation would be required.

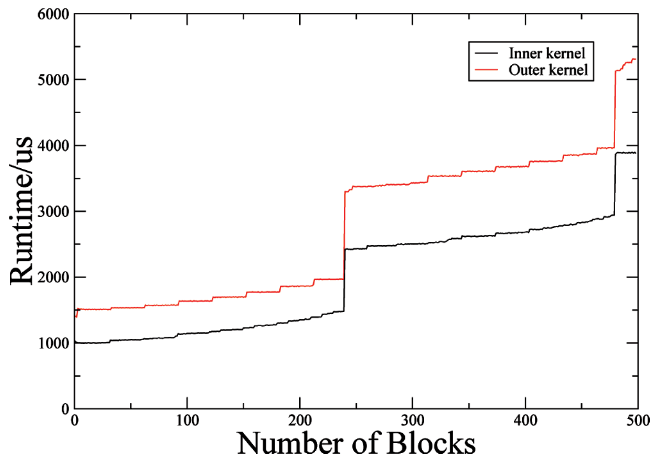

Figure 5. Run time for the nonbonded force calculation kernels on a water box as a function of the number of blocks per invocation and run on an Nvidia Tesla C1060 GPU (30 multiprocessors). The inner and outer kernels both have an occupancy of 8 blocks/multiprocessor. Blocks are distributed across multiprocessors, with the small step increases indicating an increment in the number of simultaneous blocks. The large steps indicate the device is fully populated with blocks and that some MPs must sequentially process further block. Optimal resource usage occurs immediately before these steps. The effect of gradual divergence between multiprocessors is seen as block count increases. The minimum run time for the fully parallel case would be 3.4 ms.

VII. Conclusions

We presented a molecular dynamics application, ACEMD, designed to reach the microsecond time scale even on cost-effective workstation hardware using the computational power of GPUs. It supports the CHARMM27 and Amber force field and is therefore suitable for use in modeling biomolecular systems. The ability to model these systems for tens of nanoseconds per day makes it feasible to perform simulations of up to the microsecond scale over the course of a few weeks on a suitable GPU-equipped machine. Calculations lasting a few weeks are perfectly reasonable tasks on workstation-class computers equipped with single or multiple GPUs.

ACEMD has been extensively tested since August 2008 through its deployment on the several thousand GPU-equipped PCs which participate in the volunteer distributed computing project GPUGRID.net,[49] based on the Berkeley Open Infrastructure for Networked Computing (BOINC)[50] middleware. At the time of writing, GPUGRID.net delivers over 30 Tflops of sustained performance[51] and is thus one of the largest distributed infrastructures for molecular simulations, producing thousands of nanosecond long trajectories per day for high-throughput molecular simulations, for instance, for accurate virtual screening.[52]

The current implementation of ACEMD limits its parallel performance to just 3 GPUs due to a simple task parallelization. We plan to extend the use of ACEMD on more GPUs but keeping the focus on scalability, so small numbers of GPUs (1−32). Ideally the optimal system for ACEMD would rely on a single node attached to a large number of GPUs via individual PCIe expansion slots in order to take advantage of the large interconnect bandwidth. ACEMD would potentially scale very well on such a machine due to the fact that it is entirely executing on the GPU devices, obtaining CPU loads within just 5%. For efficient scaling across a GPU-equipped cluster, we anticipate that a refactoring of the parallelization scheme to use a spatial decomposition method[4] would be necessary, moving away from the simple task parallelization used in this work. Possible future developments also include support of forthcoming programming languages for GPUs, for example, OpenCL,[53] a development library which is intended to provide a hardware-agnostic, data-parallel programming model. While GPU devices are commonly present in desktop and workstation computers for graphics purposes, as accelerator processors they have yet to become routinely integrated components of the compute cluster systems typically used for high-performance computing (HPC) systems. GPU workstations, such as the one used in this work, are readily available, while GPU clusters are slowly appearing.[57] In order to scale efficiently, low-latency, high-bandwidth communications between nodes is necessary. For example, Bowers et al.[5] describe the scaling of the Desmond MD program over an Infiniband[54] network and demonstrate improved scaling when using custom communications routines tailored to the requirements of the algorithm and the capabilities of the network technology.

Accelerated molecular dynamics on GPUs as provided by ACEMD should be of wide interest to a large number of computational scientists as it provides performance comparable to that achievable on standard CPU supercomputers in a laboratory environment. Even research groups that have routine access supercomputing time might find useful the ability to run simulations locally for longer time windows and with added flexibility.

Acknowledgment. This work was partially funded by the HPC-EUROPA project (R113-CT-2003-506079). G.G. acknowledges support from the Marie Curie Intra-European Fellowship (IEF). G.D.F. acknowledges support from the Ramon y Cajal scheme and also from the EU Virtual Physiological Human Network of Excellence. We gratefully acknowledge the advice of Sumit Gupta (Nvidia), Acellera Ltd. (http://www.acellera.com) for use of their resources, and Nvidia Corp. (http://www.nvidia.com) for their hardware donations. We thank Ignasi Buch for help in debugging the software.

References

(1) Giupponi, G.; Harvey, M. J.; de Fabritiis, G. *Drug Discovery Today* **2008**, *13*, 1052.

(2) Shaw, D. E. Anton, a Special-Purpose Machine for Molecular Dynamics Simulation. *Proceedings of the 34th annual international symposium on Computer architecture*; 2007.

(3) Bowers, K. J.; Dror, R. O.; Shaw, D. E. *J. Chem. Phys.* **2006**, *24*, 184109.

(4) Bowers, K. J.; Dror, R. O.; Shaw, D. E. *J. Phys.: Conf. Ser.* **2005**, *16*, 300–304.

(5) Bowers, K. J.; Chow, E.; Xu, H.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Moraes, M. A.; Sacerdoti, F. D.; Salmon, J. K.; Shan, Y.; Shaw, D. E. Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters. *Proceedings of ACM/IEEE Conference on SuperComputing 2006*, Tampa, FL, Nov 11−17, 2006; ACM: New York, 2006.

(6) Fitch, B.; Rayshubskiy, A.; Eleftheriou, M.; Ward, T.; Giampapa, M.; Pitman, M.; Germain, R. IBM RC23956, May 2006, *12*.

(7) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781–1802.

(8) Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. *J. Chem. Theor. Comput.* **2008**, *4*, 435–447.

(9) De Fabritiis, G. *Comput. Phys. Commun.* **2007**, *176*, 600.

(10) Luttmann, E.; Ensign, D.; Vaidyanathan, V.; Houston, M.; Rimon, N.; Oland, J.; Jayachandran, G.; Friedrichs, M.; Pande, V. *J. Comput. Chem.* **2008**, *30*, 268.

(11) Shirts, M.; Pande, V. S. *Science* **2000**, *290*, 1903–1904.

(12) Harvey, M. J.; Giupponi, G.; Villà-Freixa, J.; De Fabritiis, G. PS3GRID.NET: Building a distributed supercomputer using the PlayStation 3. *Distributed & Grid Computing-Science Made Transparent for Everyone. Principles, Applications and Supporting Communities*; in press.

(13) Anderson, D. *Bekeley Open Infrastructure for Network Computing*; http://www.boinc.berkely.edu (accessed Apr 15, 2009).

(14) *OpenGL-The Industry Standard for High Performance Graphics*; Khronos Group: http://www.khronos.org/opengl/ (accessed Apr 15, 2009).

(15) Nickolls, J.; Buck, I.; Garland, M.; Skadron, K. *ACM Queue* **2008**, *6*.

(16) Stone, J.; Phillips, J.; Freddolino, P.; Hardy, D.; Trabuco, L.; Schulten, K. *J. Comput. Chem.* **2007**, *28*, 2618–2640.

(17) van Meel, J. A.; Arnold, A.; Frenkel, D.; Portegies-Zwart, S. F.; Belleman, R. G. *Mol. Simul.* 2008.

(18) Phillips, J. C.; Stone, J. E.; Schulten, K. Adapting a message-driven parallel application to GPU-accelerated clusters. *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*; IEEE Press: 2008.

(19) Friedrichs, M. S.; Eastman, P.; Vaidyanathan, V.; Houston, M.; LeGrand, S.; Beberg, A. L.; Ensign, D. L.; Bruns, C. M.; Pande, V. S. **2009**, *30*, 864–872.

(20) Ufimtsev, I.; Martinez, T. *J. Chem. Theory Comput.* **2008**, *4*, 222–231.

(21) Anderson, A. G.; Goddard, W. A., III; Schröder, P. *Comput. Phys. Commun.* **2008**, *177*, 298–306.

(22) Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amador-Bedolla, C.; Aspuru-Guzik, A. *J. Phys. Chem. A* **2008**, *112*, 2049–2057.

(23) Yasuda, K. *J. Chem. Theor. Comput.* **2008**, *4*, 1230–1236.

(24) Ewald, P. *Ann. Phys.* **1921**, *64*, 253–287.

(25) MacKerell, A. D., Jr.; et al. *J. Phys. Chem. B* **1998**, *102*, 3586.

(26) Ponder, J. W.; Case, D. A. *Adv. Protein Chem.* **2003**, *66*, 27–85.

(27) Bonomi, M.; Branduardi, D.; Bussi, G.; Camilloni, C.; Provasi, D.; Raiteri, P.; Donadio, D.; Marinelli, F.; Pietrucci, F.; Broglia, R. A.; Parrinello, M. PLUMED: a portable plugin for free-energy calculations with molecular dynamics; 2009.

(28) Dongarra, J. J. Performance of various computers using standard linear equations software; Technical Report, Netlib report CS-89-85; 2007.

(29) Intel 64 and IA-32 Architectures Optimization Reference Manual, Document 248966-018, Technical Report; Intel: 2009.

(30) NVIDIA CUDA Compute Unified Device Architecture Programming Guide; Technical Report 2.0; NVIDIA Corp.: 2008.

(31) Essman, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G. *J. Chem. Phys.* **1995**, *19*, 8577–8593.

(32) Feenstra, K. A.; Hess, B.; Berendsen, H. J. C. *J. Comput. Chem.* **1999**, *20*, 786.

(33) Hardy D. J. *MDX libraries*; University of Illinois at Urbana-Champaign: 2007; http://www.ks.uiuc.edu/Development/MDTools/namdlite.

(34) ACEMD website; http://multiscalelab.org/acemd (accessed Apr 15, 2009).

(35) CUDA CUFFT Library; Document PG-00000-003 V2.0, Technical Report; NVIDIA Corp.: 2008.

(36) Kräutler, V.; van Gunsteren, W. F.; Hünenberger, P. H. *J. Comput. Chem.* **2001**, *22*, 501–508.

(37) Andersen, H. C. *J. Comput. Phys.* **1983**, *52*, 24–34.

(38) Verlet, L. *Part. Part. Syst. Charact.* **1967**, *159*, 98.

(39) Feenstra, K.; Hess, B.; Berendsen, H. *J. Comput. Chem.* **1999**, *20*, 786–798.

(40) Lippert, R. A.; Bowers, K. J.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I. *J. Chem. Phys.* **2007**, *126*, 046101.

(41) Tironi, I. G.; Sperb, R.; Smith, P. E.; van Gunsteren, W. F. *J. Chem. Phys.* **xxxx**, *102*, 5451–5459.

(42) Plimpton, S.; Hendrickson, B. *J. Comput. Chem.* **1996**, *17*, 326–337.

(43) Ensign, D.; Kasson, P.; Pande, V. *J. Mol. Biol.* **2007**, *374*, 806–816.

(44) Piana, S.; Laio, A.; Marinelli, F.; Van Troys, M.; Bourry, D.; Ampe, C.; Martins, J. *J. Mol. Biol.* **2008**, *375*, 460–470.

(45) Humprey, W.; Dalke, A.; Schulten, K. *J. Mol. Graphics* **1996**, *14*, 33.

(46) Buch, I.; De Fabritiis, G. Movie of Vilin unfolding: http://www.vimeo.com/2505856 (accessed Apr 15, 2009).

(47) Vision Lab, U. FASTRA GPU SuperPC: http://fastra.ua.ac.be/en/index.html (accessed Apr 15, 2009).

(48) PCI Express Base Specification; Technical Report Revision 2.0; PCI Special Interest Group, 2007.

(49) De Fabritiis, G. The GPUGRID Project homepage: http://www.gpugrid.net (accessed Apr 15, 2009).

(50) Anderson, D. L. BOINC: A System for Public-Resource Computing and Storage. *Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*; 2004.

(51) The BOINCStats Project statistics homepage: http://www.boincstats.com (accessed Apr 15, 2009).

(52) Holden, C., Ed. *Science* **2008**, *321*, 1425.

(53) OpenCL-The open standard for parallel programming of heterogeneous systems; Khronos Group: http://www.khronos.org/opengl/ (accessed Apr 15, 2009).

(54) InfiniBand Architecture Specification; Technical Report Release 1.2.1; Infiniband Trade Association: 2007. Available online at http://www.infinibandta.org/specs/register/publicspec/ (accessed Apr 15, 2009).

(55) The warp size is device specific and 32 for all current devices. As an implementation detail, each MP has 8 scalar cores and the warp threads are scheduled as 4 groups of 8 threads, with the execution of the groups pipelined on the scalar cores.

(56) This technique is also used in 18 and illustrates the general principal that, when programming the current generation of GPUs, the most efficent algorithms are those which favor simpler flow control and minimize global memory access even at the expense of redundant computation.

(57) Nvidia. Tesla-equipped Tsubame cluster; Tokyo Institute of Technology.

CT9000685