Computer Physics Communications 184 (2013) 374–380

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

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

SPFP: Speed without compromise—A mixed precision model for GPU accelerated molecular dynamics simulations

Scott Le Grand [a], Andreas W. Götz [b], Ross C. Walker [b,c,*]

[a] Amazon Web Services, 2201 Westlake Ave., Suite 500, Seattle, WA 98121, USA
[b] San Diego Supercomputer Center, University of California San Diego, 9500 Gilman Drive MC0505, La Jolla, CA 92093, USA
[c] Department of Chemistry and Biochemistry, University of California San Diego, 9500 Gilman Drive MC0505, La Jolla, CA 92093, USA

ARTICLE INFO

Article history:
Received 31 May 2012
Received in revised form
22 August 2012
Accepted 16 September 2012
Available online 25 September 2012

Keywords:
Molecular dynamics
Graphic processing unit
Accelerator
Precision model
SPSP
SPDP
DPDP
SPFP

ABSTRACT

A new precision model is proposed for the acceleration of all-atom classical molecular dynamics (MD) simulations on graphics processing units (GPUs). This precision model replaces double precision arithmetic with fixed point integer arithmetic for the accumulation of force components as compared to a previously introduced model that uses mixed single/double precision arithmetic. This significantly boosts performance on modern GPU hardware without sacrificing numerical accuracy. We present an implementation for NVIDIA GPUs of both generalized Born implicit solvent simulations as well as explicit solvent simulations using the particle mesh Ewald (PME) algorithm for long-range electrostatics using this precision model. Tests demonstrate both the performance of this implementation as well as its numerical stability for constant energy and constant temperature biomolecular MD as compared to a double precision CPU implementation and double and mixed single/double precision GPU implementations.

1. Introduction

Powerful graphics processing units (GPUs) that deliver a high peak performance of both integer and floating point arithmetics are common components of desktop workstations and are becoming increasingly ubiquitous as specialized accelerator hardware in modern high-performance computing platforms [1]. The potential of GPU hardware for an economically efficient acceleration of scientific applications has long been realized [2,3] and mature implementations are available for molecular dynamics (MD) simulations of condensed phase biomolecular systems providing capabilities to researchers that can surpass traditional CPU-based implementations [4–9]. In order to achieve high performance on GPUs, however, it is mandatory to implement algorithms that are able to exploit the specific hardware architecture of GPUs. In a recent publication [9] we introduced a GPU implementation of the AMBER [10] PMEMD MD engine that employs mixed single/double precision arithmetic (SPDP) to achieve maximum performance while maintaining numerical stability for MD simulations that is comparable to the full double precision (DPDP) GPU and CPU implementations of PMEMD.

Our original motivation behind the development of the SPDP precision model was driven by the fact that GPUs for the consumer market have traditionally had single precision floating point performance that outstrips double precision performance. In the case of NVIDIA Fermi series GPUs, this was a ratio of between 8 to 1 (GTX580) and 2 to 1 (M2090). Our SPDP model was designed to deliver optimum performance as long as this ratio was no worse than 10 to 1. Unfortunately, with the release of the Kepler I series of NVIDIA GPUs based on the GK104 chipset this performance ratio is worse than 20 to 1 for both the consumer (GTX680/690) and HPC (K10) cards. Thus, while the peak single precision performance doubled (GTX680 vs GTX580), the computational throughput that can be achieved with the SPDP precision model of the PMEMD GPU implementation decreased by approximately 33%. This was due almost entirely to the double precision performance decreasing by approximately the same amount. In order to address this we have revisited the need for double precision arithmetics with the aim to replace it in a way that has no effect on simulation accuracy while allowing us to fully exploit this new generation of GPU hardware without adversely affecting performance on previous generation hardware.

In this contribution we introduce a new precision model, termed SPFP, that combines single precision with 64-bit fixed

* Corresponding author at: San Diego Supercomputer Center, University of California San Diego, 9500 Gilman Drive MC0505, La Jolla, CA 92093, USA.
E-mail addresses: rcw@sdsc.edu, ross@rosswalker.co.uk (R.C. Walker).

point integer arithmetic in place of double precision floating point arithmetic. We present our implementation of the SPFP precision model in the AMBER PMEMD MD engine for all-atom classical MD using the AMBER [11] or CHARMM [12] pairwise additive force fields for biomolecular simulations on CUDA [13] enabled NVIDIA GPUs. We show that this implementation affords numerical stability that is equivalent to the SPDP precision model, both for generalized Born (GB) [14] implicit solvent simulations, as well as explicit solvent simulations using the particle mesh Ewald (PME) [15] algorithm while providing a considerably higher computational throughput, in particular on the new NVIDIA Kepler hardware such as the recently released GTX680 GPU. At the same time this new precision model substantially reduces the memory footprint by avoiding the need for accumulation buffers.

2. Technical details

For historical reasons all traditional CPU codes in AMBER are written entirely using double precision (DP) floating point arithmetic. For our GPU implementation of the AMBER MD engine PMEMD we distinguish three precision models that differ in the use of single precision (SP) and double precision floating point arithmetic [9]. In the SPDP precision model the real-space and PME contributions to the nonbonded forces are calculated in SP arithmetic, but bonded terms and force accumulation of the real-space part of the nonbonded forces are calculated in double precision. Alternatively, everything is computed and accumulated in single precision (SPSP) or double precision (DPDP). The SHAKE algorithm [16,17] is implemented in DP for all precision models to maintain numerical stability. The aim in developing the SPDP model for the GPU implementation of PMEMD was to achieve numerical stability during MD simulations equivalent to that of traditional DP implementations but with a performance as close to the SPSP model as possible.

Narumi et al. [18] studied the use of various tricks to approximate double precision using combinations of single precision floating point and integer arithmetic. Specifically, they proposed the construction of a quasi double-precision structure built from a 32-bit integer and a single precision float. This approach provides approximately 48-bit floating point precision without requiring double precision arithmetic at the cost of 5 single precision operations per extended addition and the use of additional temporary registers. We initially investigated the use of this approach to improve performance on GTX680 cards, however, since this adds only approximately 2 significant figures to the force accumulation step the effective precision is lower than in our SPDP model. The motivation behind the approach of Narumi et al. was to provide increased precision at a time when GPUs supported only single precision arithmetic.

All NVIDIA GPUs since the release of the Tesla C1060 have supported double precision and more importantly the ability to directly cast a 32-bit float to a 64-bit integer. This functionality coupled with our desire to maintain the precision of our SPDP model motivated us to instead develop a new precision model, termed SPFP, that is equivalent to the SPDP precision model but replaces all DP arithmetic with 64-bit fixed point (FP) integer arithmetic. In addition the PME forces resulting from the reciprocal sum charge grid are now summed in 64-bit FP integer arithmetic as opposed to SP floating point arithmetic in the SPDP model.

2.1. Fixed precision

The relative precision of a number x expressed as a floating point value $\mathrm{fl}_n(x)$ with n bits for the significand is given as

$$\left|\frac{\mathrm{fl}_n(x) - x}{x}\right| < 2^{-n}. \tag{1}$$

For an IEEE754 double precision number with 53 bits for the significand including a hidden bit this corresponds to a relative precision of $\approx 10^{-16}$. This precision and the dynamic range offered by 64-bit double precision floating point numbers is not required for typical MD simulations. For the reasons discussed below we assert that fixed point 64-bit integer representations Q$m.f$ with an appropriate choice of magnitude bits m and fractional bits f can be used instead.

Specifically our SPFP precision model replaces the double precision force accumulation with Q24.40 fixed point integer accumulation while energy and virial accumulation is carried out using Q34.30 fixed point integers. This provides 7 significant figures to the left of the radix point and 12 to the right for the forces. Given that the forces typically never exceed numerical values of 100.0 in internal units this provides more than enough range for all situations that would be encountered in a stable MD simulation. While the forces are vectors of length $3 \times N_{\mathrm{atom}}$, where N_{atom} is the number of atoms, the energy is scalar with only a single accumulator. This means that for a large system the energy values could overflow a Q24.40 fixed point accumulator. However, energies have no effect on the trajectory and are typically only written to 4 decimal places and thus we use Q34.30 FP arithmetic which provides approximately 10 significant figures to the left of the radix point and 9 to the right. C/CUDA/PTX assembly code for scaling the 32-bit SP force and energy components and conversion to 64-bit FP representation is given in Fig. 1.

A key component of our original implementation is deterministic operation. This was achieved by using output buffers for each force vector component which allowed accumulation to be carried out in the same order even when running in parallel. We have retained this in our SPFP implementation but with significant memory savings by exploiting the fact that while floating-point math is not associative, integer math is. One can therefore use 64-bit atomic operations to accumulate the forces and the reciprocal sum charge grid in a deterministic fashion. The gain in doing this on the Fermi GPUs (for example GTX580) is marginal since atomic operations are not particularly fast. However, the GTX680 (Kepler, GK104) series of GPUs includes atomic operation hardware that is approximately 3x faster than the previous generation. This makes the use of atomic operations attractive and ultimately provides a speed benefit over the Fermi series GPUs. Beyond the introduction of fixed point accumulation the code is otherwise the same with the exception that the reciprocal sum charge grid is now summed in Q24.40 fixed precision as opposed to 32-bit floating point. This ultimately provides an additional improvement in achievable precision.

2.2. Memory requirements

The system size limitation of our SPDP (and DPDP) precision model implementations is dominated by the memory requirements of the output buffers for the real-space nonbonded interactions. In the case of GB simulations that do not use a cutoff for real-space nonbonded interactions, a total of $N \times (N/W + 1)$ output buffers are required to achieve high computational performance while avoiding race conditions [9], where N is the total number of atoms and W is the warp size[1]. Much larger numbers of atoms can be handled with the PME implementation [19] since long-range electrostatics beyond a cutoff are handled with the PME algorithm. Hence, the number of real-space nonbonded interactions for each

[1] A warp is a subgroup of a CUDA thread block containing 32 threads that execute synchronously [13]. The size of a warp is subject to change without warning between hardware generations.

```
#define FORCESCALEF  (float)(1ll << 40)
#define ENERGYSCALEF (float)(1ll << 30)

f_fp = lliroundf(FORCESCALEF * f);
e_fp = lliroundf(ENERGYSCALEF * e);

__device__ inline long long int lliroundf(float f)
{
  long long int l;
  asm("cvt.rni.s64.f32     %0, %1;" : "=l"(l) : "f"(f));
  return l;
}
```

Fig. 1. C/CUDA/PTX assembly code for scaling 32-bit single precision force components `f` and energy components `e` to Q24.40 and Q34.30 fixed precision representations `f_fp` and `e_fp`, respectively, as implemented in the CUDA version of the AMBER MD engine PMEMD. `cvt.rni.s64.f32` casts a floating-point value to a 64-bit signed integer, rounding to the nearest integer. Inline PTX assembly code was required to precisely describe the intent of this type conversion.

Table 1
Approximate maximum atom counts that can be treated with the GPU implementation of GB implicit solvent and PME explicit solvent simulations in a development version of AMBER 13 using the SPDP and SPFP precision models. Test systems are droplets and cubic boxes of TIP3P water molecules, respectively (for details on the simulations see the text). Error-correction code (ECC) was switched off on the Tesla card.

GPU		GB (max atoms)		PME (max atoms)	
Type	Memory (GB)	SPDP	SPFP	SPDP	SPFP
GTX580	3.0	39,250	1,195,000	1,060,000	1,240,000
Tesla M2090	6.0	54,000	1,740,000	2,230,000	2,680,000
GTX680	2.0	32,000	960,000	710,000	920,000

atom is constant for PME and the memory requirement scales only linearly instead of quadratically with the number of atoms.

Our implementation of the SPFP precision model does not use these output buffers. Instead, we are using 64-bit atomic operations to accumulate the forces and thus do not require the accumulation buffers since the order of summation no longer leads to rounding differences. In this way we preserve deterministic operation but without the excessive memory overhead. We exploit the hardware support of atomic operations provided by the GPU. This allows us to use a fire and forget approach to the accumulations.

An added benefit to this approach is that we now also have deterministic operation in parallel irrespective of the number of MPI tasks. Previously deterministic operation was only guaranteed for a constant number of MPI tasks using a static allocation of work. Our SPFP approach allows work to be dynamically load balanced between MPI tasks but with deterministic execution still being preserved. A potential downside of this approach is that the system size is now ultimately limited by the fixed precision. This limitation arises because the energy increases linearly with the number of atoms and eventually will overflow the Q34.30 fixed precision accumulator. The potential energy per atom in simulations with commonly used AMBER force fields typically does not exceed 10 kcal/mol. With approximately 10 significant figures to the left of the radix point, this means that it is safe to assume that the limit for overflow is well past 10 million atoms which is sufficient to cover all simulations currently run with the AMBER software.

The software base used for all simulations in this work was a development version of AMBER 13. The executables were built under the RedHat Enterprise Linux 6 operating system with Intel compiler version 11.1.069 and the NVIDIA CUDA compiler version 4.2. The performance tests were run on machines equipped with an 8-core AMD FX-8150 processor clocked at 3.6 GHz with the GPU cards connected to a PCIe x16 slot and using the NVIDIA Linux Driver version 295.41.

Table 1 shows the maximum system size that can be treated with the GPU implementation using the SPDP and SPFP precision models for GB and PME simulations as a function of GPU hardware. Test systems are droplets and cubic boxes of TIP3P water molecules, respectively. All simulations were performed without temperature control using a time step of 2 fs and the SHAKE algorithm to constrain bonds to hydrogen atoms. The Hawkins, Cramer, Truhlar GB model [20] was used without cutoff for the nonbonded interactions and a cutoff value of 15 Å for GB radii. For the PME simulations, the default cutoff of 8 Å was used for the nonbonded interactions.

For GB simulations, the SPFP implementation effectively removes the system size limitation due to available device memory. Simulations of approximately one million atoms become possible with only 2 gigabyte (GB) of GPU memory. Meaningful simulations of such large systems, however, will require hundreds of nanoseconds of simulation time or more to provide for proper sampling. Future improvements will have to focus on further speedups to provide the corresponding computational throughput.

In the case of PME simulations the difference in maximum atom counts that can be treated with the different GPU implementations is not as dramatic as is the case for GB simulations. This is because the number of nonbonded interactions that are treated in real-space (within a pre-set cutoff) scales linearly with the system size as opposed to quadratic. Over 900,000 atoms can be handled with memory as low as 2 GB (GTX680) and more than 2,600,000 atoms if 6 GB of memory is available (M2090).

2.3. Performance

Table 2 shows the performance in terms of computational throughput that can be achieved with the SPDP, SPFP as well as DPDP precision models as a function of GPU hardware. An implicit solvent GB simulation of apo-myoglobin (2492 atoms) and an explicit solvent PME simulation of dihydrofolate reductase in a rectangular box of TIP3P [21] water atoms (DHFR, 23,558 atoms including solvent) using the ff99SB [22] version of the AMBER force field were chosen as representative examples of typical research scenarios. Details of the simulations are as described above for the water droplets/boxes. Input files for these simulations are provided in the Supporting Information.

Table 2
Throughput timings (ns/d) for AMBER GB simulations of apo-myoglobin (2492 atoms) and AMBER PME simulations of DHFR (23,558 atoms) with a time step of 2 fs using the serial GPU version with different precision models. For details on the simulations and the hardware and software stack, see text.

Precision model	GB (apo-myoglobin)			PME (DHFR)		
	SPFP	SPDP	DPDP	SPFP	SPDP	DPDP
GTX580	106.2	94.7	17.0	54.8	53.0	10.0
M2090	89.5	84.8	28.7	46.6	46.2	16.2
GTX680	155.8	85.0	11.4	74.4	45.9	6.6

We first notice that the SPFP precision model outperforms the SPDP and DPDP precision models on all tested hardware, both for GB and PME simulations. The SPFP precision model is able to fully exploit the higher single precision performance of the new NVIDIA Kepler I (GTX680) hardware while both the SPDP and in particular the DPDP precision model suffer from a drop in performance on this hardware as compared to the older Fermi hardware (GTX580, M2090). Compared to the SPDP precision model, the computational throughput that can be achieved with the SPFP precision model increases by up to 12% on Fermi hardware and 83% and 62% for GB and PME simulations, respectively, on Kepler I hardware. We obtain benchmark throughputs of nearly 156 ns/d for GB simulations of apo-myoglobin and 75 ns/d for PME simulations of DHFR.

3. Numerical stability

In what follows we will demonstrate the numerical accuracy of our SPFP GPU implementation in comparison to the SPDP GPU implementation and the full double precision GPU (DPDP) and CPU implementations. In addition to apo-myoglobin described above, we have used TRPCage [23] (304 atoms), ubiquitin [24,25] (1231 atoms, PDB code 1UBQ), and nucleosome (25,095 atoms, PDB code 1KX5) as test systems for implicit solvent GB simulations and dihydrofolate reductase (DHFR, 23,558 atoms), FactorIX (90,906 atoms), Cellulose (408,576 atoms) and the satellite tobacco mosaic virus (STMV, 1,066,846 atoms) as test systems for explicit solvent PME simulations. Input files for all simulations are provided in the Supporting Information.

3.1. Force errors

Tables 3 and 4 show the effect of the numerical precision model on the accuracy of force calculations as compared to the CPU implementation as the reference precision model for implicit solvent GB and explicit solvent PME calculations, respectively, with settings as described above for the water droplets/boxes.

For GB simulations, the DPDP model matches the reference forces very closely with maximum deviations not exceeding 10^{-7} kcal/(mol Å) and root-mean-square (RMS) deviations not exceeding 10^{-8} kcal/(mol Å), even for systems as large as nucleosome. These deviations are entirely due to the different order of execution of the floating point operations in the CPU and GPU implementations. The forces obtained from the SPFP precision model show deviations that are much larger but identical to the SPDP precision model. For GB simulations we have already shown [9] that these errors are sufficiently small such that the SPDP GPU implementation generates trajectories with numerical stability that is equivalent to the CPU implementation. It is thus to be expected that numerically stable simulations are also possible with the SPFP precision model.

We observe similar force errors for explicit solvent PME simulations. With the exception of FactorIX, maximum deviations with respect to the reference CPU implementation do not exceed 10^{-7} kcal/(mol Å) and RMS deviations do not exceed 10^{-8} kcal/(mol Å). The deviations for FactorIX are slightly higher, however these can be reduced by increasing the FFT grid dimensions from their default value (not shown). The force errors of the SPFP precision model and the SPDP precision model are identical, consistently small and comparable to the deviations of the GB forces. A thorough analysis showing that numerically stable MD simulations using the PME algorithm are possible with the SPDP and SPFP precision models will be presented elsewhere [19].

3.2. Energy conservation

A numerically stable implementation of MD should be able to conserve energy during constant energy simulations. To this end we have performed constant energy MD simulations of TRPCage (100 ns), ubiquitin (50 ns) and apo-myoglobin (20 ns) after an

Table 3
Deviations of forces (in kcal/(mol Å)) of the AMBER PMEMD GPU implementation using different precision models for GB implicit solvent simulations as compared to reference values obtained with the CPU implementation.

Precision model	TRPCage (304 atoms)	ubiquitin (1231 atoms)	apo-myoglobin (2492 atoms)	nucleosome (25,095 atoms)
max deviation				
SPFP	5.6×10^{-5}	3.7×10^{-4}	1.6×10^{-4}	1.1×10^{-3}
SPDP	5.6×10^{-5}	3.7×10^{-4}	1.6×10^{-4}	1.1×10^{-3}
DPDP	1.1×10^{-8}	7.3×10^{-8}	3.4×10^{-8}	8.0×10^{-8}
RMS deviation				
SPFP	7.0×10^{-6}	1.5×10^{-5}	8.1×10^{-6}	3.0×10^{-5}
SPDP	7.0×10^{-6}	1.5×10^{-5}	8.1×10^{-6}	3.0×10^{-5}
DPDP	1.5×10^{-9}	3.6×10^{-9}	2.6×10^{-9}	3.2×10^{-9}

Table 4
Deviations of forces (in kcal/(mol Å)) of the AMBER PMEMD GPU implementation using different precision models for PME explicit solvent simulations as compared to reference values obtained with the CPU implementation.

Precision model	DHFR (23,558 atoms)	FactorIX (90,906 atoms)	Cellulose (408,576 atoms)	STMV (1,066,846 atoms)
max deviation				
SPFP	3.4×10^{-4}	2.2×10^{-3}	1.9×10^{-3}	4.0×10^{-3}
SPDP	3.4×10^{-4}	2.2×10^{-3}	1.9×10^{-3}	4.0×10^{-3}
DPDP	6.0×10^{-8}	2.1×10^{-6}	4.6×10^{-8}	9.4×10^{-8}
RMS deviation				
SPFP	2.3×10^{-5}	1.2×10^{-4}	7.8×10^{-5}	1.3×10^{-4}
SPDP	2.3×10^{-5}	1.2×10^{-4}	7.8×10^{-5}	1.3×10^{-4}
DPDP	1.5×10^{-9}	2.1×10^{-7}	1.3×10^{-9}	2.1×10^{-9}

Table 5
Energy drifts per degree of freedom (kT/ns/dof) from GB implicit solvent simulations of 100 ns (TRPCage), 50 ns (ubiquitin) and 20 ns (apo-Myoglobin). The SHAKE algorithm to constrain bond lengths to hydrogen atoms was used for a time step of 2.0 fs, no constraints were used for smaller time steps.

Time step	0.5 fs	1.0 fs	2.0 fs
TRPCage (304 *atoms*)			
CPU	0.000006	0.000066	0.000355
GPU (DPDP)	0.000012	0.000082	0.000382
GPU (SPDP)	0.000003	0.000070	0.000222
GPU (SPFP)	0.000011	0.000066	0.000355
ubiquitin (1231 *atoms*)			
CPU	0.000004	0.000011	−0.000216
GPU (DPDP)	0.000001	0.000006	−0.000247
GPU (SPDP)	0.000003	0.000030	−0.000165
GPU (SPFP)	−0.000003	0.000006	−0.000350
apo-myoglobin (2492 *atoms*)			
CPU	0.000012	0.000094	0.000416
GPU (DPDP)	−0.000004	0.000117	0.000290
GPU (SPDP)	0.000019	0.000185	0.000139
GPU (SPFP)	−0.000004	0.000122	0.000254

initial equilibration for 1 ns at 300 K using Langevin dynamics. Details of the simulations are as described above for the water droplets/boxes. The center of mass motion was removed before starting the constant energy runs. We investigated simulations using time steps of 0.5 fs and 1.0 fs without constraints as well as a time step of 2.0 fs with bonds to hydrogen atoms constrained using the SHAKE algorithm with a relative geometrical tolerance of 10^{-6} Å. We restrict ourselves to an analysis of the SPFP precision model in comparison to published results [9] for the SPDP and DPDP precision models as well as the double precision CPU implementation for implicit solvent GB simulations. Input files for these simulations are provided in the Supporting Information. A detailed discussion of results for explicit solvent PME simulations are presented in another publication [19].

Table 5 summarizes the energy drifts in kT per degree of freedom (dof) while Fig. 2 shows a plot of the total energy for the trajectories with the different precision models at a time step of 0.5 fs. Plots for the larger time steps are shown in the Supporting Information. The SPFP precision model is able to conserve energy to the same degree as the other precision models and the reference CPU implementation, making numerically stable simulations possible.

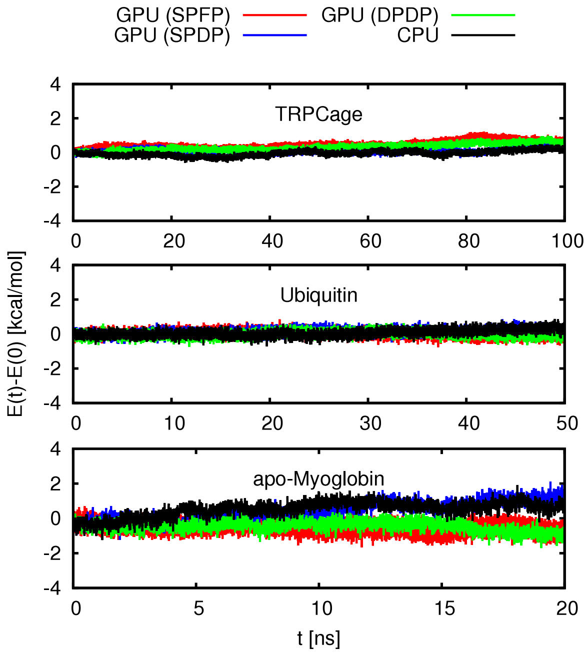

Fig. 2. Total energy (kcal/mol) along constant energy trajectories using a time step of 0.5 fs without constraints. Shown are results for TRPCage (top), ubiquitin (center) and apo-myoglobin (bottom) for different precision models.

3.3. Simulation stability

We have verified that simulations are indeed numerically stable by performing a series of 50 independent MD simulations of ubiquitin with the SPFP precision model and comparing the trajectories with published results [9] for the other precision models of our GPU implementation as well as the double precision CPU implementation. We analyze root-mean-square deviations (RMSDs) and root-mean-square fluctuations (RMSFs) of the C_α backbone carbon atoms with respect to the crystal structure (PDB code 1UBQ [24,25]). The highly flexible end tail of ubiquitin (residues 71–76) was excluded from our analysis. GB trajectories of 100 ns length using a time step of 2.0 fs were generated with settings as described above for the water droplets, however, using the Berendsen weak coupling thermostat [26] with a target temperature of 300 K and a time constant of $\tau_T = 10.0$ ps for the heat bath coupling. The initial coordinates and velocities that form the starting point of the 50 trajectories were generated using the CPU implementation using a protocol as published [9]. Input files for these simulations are provided in the Supporting Information.

Fig. 3 contains plots of the RMSD vs time. It shows that MD simulations using the GPU implementation with the SPFP precision model have the same numerical stability as GPU based simulations using the SPDP and DPDP precision models or the double precision CPU implementation. The C_α backbone carbon atoms remain within 3–4 Å of the crystal structure for all 50 independent simulations over the whole 100 ns trajectory. We have shown earlier [9], that this is not the case for an implementation that employs single precision floating point arithmetics throughout since in this case the numerical noise due to rounding errors can lead to an accumulation of errors such that ubiquitin starts to unfold. This is not the case for the SPFP precision model which can be employed for numerically stable long-timescale MD simulations.

Fig. 4 shows the RMSF values for each residue of the 50 native-state simulations. An excellent agreement is found between all precision models of the GPU implementation and the double precision CPU implementation. The majority of the residues remain within 2 to 3 Å of the experimental structure with somewhat larger fluctuations around residues 33, 48 and 63. No major structural change is taking place during the simulations which confirms that the protein remains within a native-state ensemble. We thus recommend the GPU implementation of PMEMD to be used with the SPFP precision model to reduce the computational cost compared to the SPDP and DPDP precision models while maintaining numerical stability that is equivalent to the reference double precision CPU implementation.

4. Concluding remarks

We have implemented a new precision model (SPFP) for efficient MD simulations on GPU hardware that supports rapid atomic operations. This model uses a combination of Q24.40 and Q34.30 fixed arithmetic for accumulation of forces and energies respectively to deliver numerical precision that is equivalent to our previously developed hybrid single/double precision (SPDP) model but with performance on the latest generation of NVIDIA GPUs (Kepler I, GK104) that is 60–80 % faster than our original SPDP model. There are no discernible differences in results from our previously validated SPDP model and performance on previous generation hardware (Fermi series) is the same if not slightly better

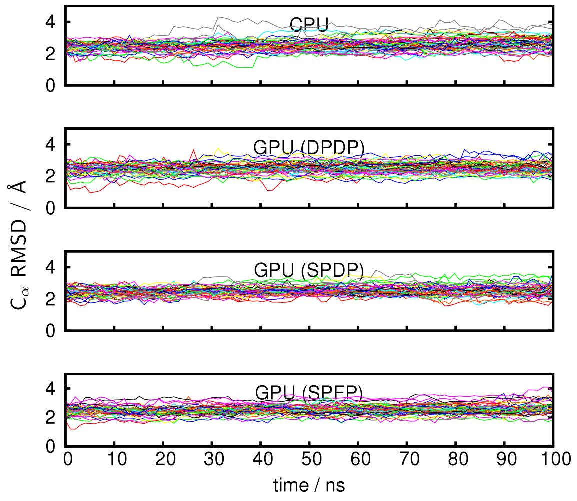

Fig. 3. Root-mean-square deviations (RMSDs) of the C_α backbone carbon atoms of ubiquitin (excluding the flexible tail, residues 71–76) with respect to the crystal structure for 50 independent trajectories as obtained with the CPU implementation and the GPU implementation of PMEMD using different precision models.

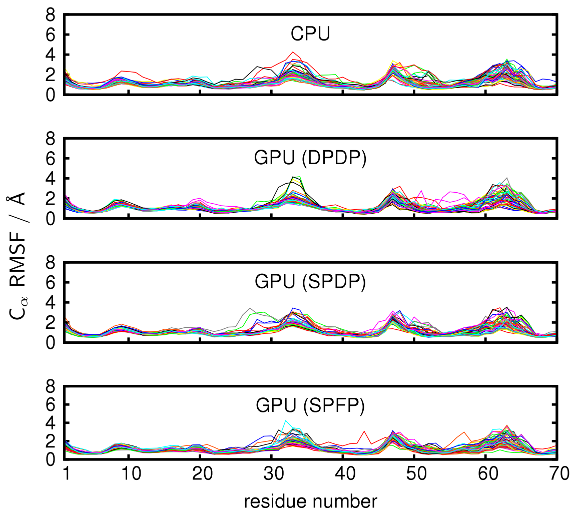

Fig. 4. Root-mean-square fluctuations (RMSFs) of the C_α backbone carbon atoms of ubiquitin residues 71–76 with respect to the crystal structure for 50 independent trajectories of 100 ns length as obtained with the CPU implementation and the GPU implementation of PMEMD using different precision models.

than the SPDP approach. At the same time, however, the memory footprint has been reduced, significantly in the case of implicit solvent GB simulations. It is our intention to release a patch, that will change the default precision mode from SPDP to SPFP for the currently available version 12 of the AMBER software shortly.

Acknowledgments

This work was funded in part by the National Science Foundation through the Scientific Software Innovations Institutes program- NSF SI2-SSE (NSF1047875 and NSF1148276) grants to R.C.W and also by the University of California (UC Lab 09-LR-06-117792) grant to R.C.W. Computer time was provided by the San Diego Supercomputer Center through National Science Foundation award TG-MCB090110 to R.C.W. and A.W.G. The work was also supported by CUDA fellowships to R.C.W. and S.LG. from NVIDIA. S.LG. would like to thank Amazon Web Services for support and also his late father (Donald George Le Grand) for inspiration.

Appendix. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.cpc.2012.09.022.

References

[1] J.S. Vetter, R. Glassbrook, J. Dongarra, K. Schwan, B. Loftis, S. McNally, J. Meredith, J. Rogers, P. Roth, K. Spafford, S. Yalamanchili, Keeneland: Bringing heterogeneous GPU computing to the computational science community, Comput. Sci. Eng. 13 (5) (2011) 90–95.

[2] A.W. Götz, T.M. Wölfle, R.C. Walker, Quantum chemistry on graphics processing units, in: C. Simmerling (Ed.), Annual Reports in Computational Chemistry, vol. 6, Elsevier, 2010, pp. 21–35. http://dx.doi.org/10.1016/S1574-1400(10)06002-0. Chapter 2.

[3] D. Xu, M.J. Williamson, R.C. Walker, Advancements in molecular dynamics simulations of biomolecules on graphical processing units, in: C. Simmerling (Ed.), Annual Reports in Computational Chemistry, vol. 6, Elsevier, 2010, pp. 21–35. http://dx.doi.org/10.1016/S1574-1400(10)06001-9. Chapter 1.

[4] M.S. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A.L. Beberg, D.L. Ensign, C.M. Bruns, V.S. Pande, Accelerating molecular dynamic simulation on graphics processing units, J. Comput. Chem. 30 (6) (2009) 864–872. http://dx.doi.org/10.1002/jcc.21209.

[5] P. Eastman, V.S. Pande, Efficient nonbonded interactions for molecular dynamics on a graphics processing unit, J. Comput. Chem. 31 (6) (2010) 1268–1272. http://dx.doi.org/10.1002/jcc.21413.

[6] M.J. Harvey, G. Giupponi, G.D. Fabritiis, ACEMD: Accelerating biomolecular dynamics in the microsecond time scale, J. Chem. Theory Comput. 5 (6) (2009) 1632–1639. http://dx.doi.org/10.1021/ct9000685.

[7] M.J. Harvey, G.D. Fabritiis, An implementation of the smooth particle mesh Ewald method on GPU hardware, J. Chem. Theory Comput. 5 (9) (2009) 2371–2377. http://dx.doi.org/10.1021/ct900275y.

[8] W.M. Brown, P. Wang, S.J. Plimpton, A.N. Tharrington, Implementing molecular dynamics on hybrid high performance computers - short range forces, Comput. Phys. Comm. 182 (4) (2011) 898–911. http://dx.doi.org/10.1016/j.cpc.2010.12.021.

[9] A.W. Götz, M.J. Williamson, D. Xu, D. Poole, S. Le Grand, R.C. Walker, Routine microsecond molecular dynamics simulations with AMBER on GPUs. 1. Generalized Born, J. Chem. Theory Comput. 8 (5) (2012) 1542–1555. http://dx.doi.org/10.1021/ct200909j.

[10] D.A. Case, T.A. Darden, T.E. Cheatham, III, C. Simmerling, J. Wang, R.E. Duke, R. Luo, R.C. Walker, W. Zhang, K.M. Merz, B. Roberts, S. Hayik, A. Roitberg, G. Seabra, J. Swails, A.W. Götz, I. Kolossváry, K.F. Wong, F. Paesani, J. Vanicek, J. Liu, X. Wu, S.R. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, G. Cui, D. Roe, D. Mathews, M. Seetin, R. Salomon-Ferrer, C. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, P. Kollman, AMBER 12, University of California, San Francisco, 2012.

[11] W.D. Cornell, P. Cieplak, C.I. Bayly, I.R. Gould, K.M. Merz, D.M. Ferguson, D.C. Spellmeyer, T. Fox, J.W. Caldwell, P.A. Kollman, A second generation force field for the simulation of proteins, nucleic acids, and organic molecules, J. Am. Chem. Soc. 117 (19) (1995) 5179–5197. http://dx.doi.org/10.1021/ja00124a002.

[12] A.D. MacKerell Jr., D. Bashford, M. Bellott, R.L. Dunbrack Jr., J.D. Evanseck, M.J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F.T.K. Lau, C. Mattos, S. Michnick, T. Ngo, D.T. Nguyen, B. Prodhom, W.E. Reiher, III, B. Roux, M. Schlenkrich, J.C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin, M. Karplus, All-atom empirical potential for molecular modeling and dynamics studies of proteins, J. Phys. Chem. B 102 (18) (1998) 3586–3616. http://dx.doi.org/10.1021/jp973084f.

[13] D.B. Kirk, W.W. Hwu, Programming Massively Parallel Processors, Morgan Kaufmann Publishers, 2010.

[14] W.C. Still, A. Tempczyk, R.C. Hawley, T. Hendrickson, Semianalytical treatment of solvation for molecular mechanics and dynamics, J. Am. Chem. Soc. 112 (16) (1990) 6127–6129. http://dx.doi.org/10.1021/ja00172a038.

[15] T. Darden, D. York, L. Pedersen, Particle mesh Ewald: an Nlog(N) method for Ewald sums in large systems, J. Chem. Phys. 98 (1993) 10090–10092. http://dx.doi.org/10.1063/1.464397.

[16] J.-P. Ryckaert, G. Ciccotti, H.J. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes, J. Comput. Phys. 23 (3) (1977) 327–341. http://dx.doi.org/10.1016/0021-9991(77)90098-5.

[17] S. Miyamoto, P.A. Kollman, Settle: An analytical version of the SHAKE and RATTLE algorithm for rigid water models, J. Comput. Chem. 13 (8) (1992) 952–962. http://dx.doi.org/10.1002/jcc.540130805.

[18] T. Narumi, T. Hamada, K. Nitadori, R. Sakamaki, S. Kameoka, K. Yasuoka, High-performance quasi double-precision method using single-precision hardware for molecular dynamics simulations with GPUs, in: Proceedings of HPC Asia APAN, 2009, pp. 160–167. http://dx.doi.org/10.1142/S0219876211002708.

[19] A.W. Götz, R. Salomon-Ferrer, D. Poole, S. Le Grand, R.C. Walker, Routine microsecond molecular dynamics simulations with AMBER – Part II: Particle mesh Ewald (2012) (in preparation).

[20] G.D. Hawkins, C.J. Cramer, D.G. Truhlar, Parametrized models of aqueous free energies of solvation based on pairwise descreening of solute atomic charges from a dielectric medium, J. Phys. Chem. 100 (51) (1996) 19824–19839. http://dx.doi.org/10.1021/jp961710n.

[21] W.L. Jorgensen, J. Chandrasekhar, J.D. Madura, R.W. Impey, M.L. Klein, Comparison of simple potential functions for simulating liquid water, J. Chem. Phys. 79 (2) (1983) 926. http://dx.doi.org/10.1063/1.445869.

[22] V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, C. Simmerling, Comparison of multiple amber force fields and development of improved protein backbone parameters, Proteins 65 (3) (2006) 712–725. http://dx.doi.org/10.1002/prot.21123.

[23] C. Simmerling, B. Strockbine, A.E. Roitberg, All-atom structure prediction and folding simulations of a stable protein, J. Am. Chem. Soc. 124 (38) (2002) 11258–11259. http://dx.doi.org/10.1021/ja0273851.

[24] S. Vijaykumar, C.E. Bugg, K.D. Wilkinson, W.J. Cook, 3-dimensional structure of ubiquitin at 2.8 å resolution, Proc. Natl. Acad. Sci. 82 (1985) 3582–3585.

[25] S. Vijaykumar, C.E. Bugg, W.J. Cook, Structure of ubiquitin refined at 1.8 å resolution, J. Mol. Bio. 194 (1987) 531–544.

[26] H.J.C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, A. DiNola, J.R. Haak, Molecular dynamics with coupling to an external bath, J. Chem. Phys. 81 (8) (1984) 3684–3690. http://dx.doi.org/10.1063/1.448118.