Supporting information for: Routine microsecond molecular dynamics simulations with AMBER on GPUs – Part I: Generalized Born

Andreas W. Götz,[†] Mark J. Williamson,[†,§] Dong Xu,[†,‖] Duncan Poole,[‡] Scott Le Grand,[‡] and Ross C. Walker[*,†,¶]

San Diego Supercomputer Center, University of California San Diego, 9500 Gilman Drive MC0505, La Jolla, CA 92093, USA, NVIDIA Corporation, 2701 San Tomas Expressway, Santa Clara, CA 95050, USA, and Department of Chemistry and Biochemistry, University of California San Diego, 9500 Gilman Drive MC0505, La Jolla, CA 92093, USA

E-mail: ross@rosswalker.co.uk

The following contains a list of CUDA GPU synchronization routines and a description of the supplementary material we are providing for download in the accompanying zip file. We also show additional plots of energy conservation during constant energy MD simulations and the radius of gyration R_g of ubiquitin during constant temperature simulations obtained with the CPU implementation and GPU implementation of PMEMD.

[*]To whom correspondence should be addressed

[†]SDSC

[‡]NVIDIA

[¶]UCSD Department of Chemistry and Biochemistry

[§]Current address: Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge, CB2 1EW, UK

[‖]Current address: Department of Chemistry and Biochemistry, Boise State University, 1910 University Drive, Boise, ID 83725-1520, USA

1 GPU synchronization routines

In order to make it as simple as possible for users to add new features to the code we created a series of GPU synchronization routines which provide an abstract way to copy relevant data such as atomic velocities, GB radii, coordinates, masses and forces to and from the GPU memory. These include for uploads:

```
gpu_upload_vel(atm_vel)
gpu_upload_rborn(atm_gb_radii)
gpu_upload_fs(atm_gb_fs)
gpu_upload_crd(atm_crd)
gpu_upload_charges(atm_qterm)
gpu_upload_masses(atm_mass)
gpu_upload_frc(atm_frc)
gpu_upload_last_vel(atm_last_vel)
```

and for downloads

```
gpu_download_frc(frc)
gpu_download_vel(vel)
gpu_download_crd(crd)
```

For performance we have implemented the entire MD algorithm on the GPU which means uploads (to GPU memory) are only needed at the beginning of a run and downloads (to CPU memory) are only needed when I/O is required, for example downloading the coordinates to write to the trajectory file. It remains nevertheless simple to add new features to the code. For example, suppose one wanted to replace the Anderson thermostat in the code with a new thermostat. The code at present is as follows:

```
#ifdef CUDA
   call gpu_vrand_reset_velocities(temp0 * factt, half_dtx)
```

```
#else
   call vrand_set_velocities(atm_cnt, vel, &
                             atm_mass_inv, temp0 * factt)
#endif
```

If you were to replace the function `vrand_set_velocities` with a new routine called `my_thermostat` for which the GPU version of the code had not yet been written it would be necessary to first download the velocities (required by the CPU thermostat routine) from the GPU, modify them on the CPU and then upload them to the GPU again once the function is complete, for example

```
#ifdef CUDA
   call gpu_download_vel(vel)
#endif
   call my_thermostat(vel,...)
#ifdef CUDA
   call gpu_upload_vel(vel)
#endif
```

This implementation will then work for both the CPU and GPU versions of the code. Of course for the GPU version this comes with the price of transferring an array with dimension three times the number of atoms between the GPU and CPU every time this function is called. If this occurs on every MD step, it would likely limit the performance of the GPU code. However, it does provide a mechanism by which new features can be easily added and tested. Once the new thermostat is tested one could then write a `gpu_my_thermostat` kernel for the GPU and remove the download and upload calls.

2 Performance measurement input files

AMBER input files used for the performance tests of the PMEMD GPU implementation (TRPCage, ubiquitin, myoglobin and nucleosome) are contained in subdirectory “Performance Measurement Inputs”. For each simulation this includes the AMBER MD input file “mdin”, the parameter and topology file “prmtop” and the starting coordinate file “inpcrd”. For convenience, the starting geometries are also included as pdb file “inpcrd.pdb”.

3 Validation: Single point forces input files

AMBER input files used for the tests of the precision of the single point forces of the PMEMD GPU implementation (TRPCage, ubiquitin, apo-myoglobin and nucleosome) are contained in subdirectory “Validation Single Point Forces Inputs”. For each simulation this includes the AMBER MD input file “mdin”, the parameter and topology file “prmtop” and the starting coordinate file “inpcrd”. For convenience, the starting geometries are also included as pdb file “inpcrd.pdb”.

4 Validation: Energy conservation input files

AMBER input files used for the energy conservation tests of the PMEMD GPU implementation (TRPCage, ubiquitin and apo-myoglobin) are contained in subdirectory “Validation Energy Conservation Inputs”. For each simulation this includes the AMBER MD input files “mdin”, the parameter and topology file “prmtop” and the starting coordinate file “inpcrd”. For convenience, the starting geometries are also included as pdb file “inpcrd.pdb”. We are including “mdin” files for all time steps (0.5 fs and 1.0 fs without constraints and 2.0 fs using SHAKE constraints for bonds to hydrogen atoms). In all cases, first an equilibration was performed (equil) followed by removal of the center of mass motion (removeCOM) before running the constant energy validation simulations (NVE).

5 Validation: Energy conservation plots

Plots showing the total energy for the trajectories of TRPCage, ubiquitin and apo-myoglobin for the different precision models and time steps $dt = 0.5\,\text{fs}$, $dt = 1.0\,\text{fs}$ and $dt = 2.0\,\text{fs}$, are shown in Figure 1 to Figure 3.

6 Validation: Structural Properties input files

AMBER input files used for the analysis of structural properties of ubiquitin are contained in subdirectory “Validation Structural Properties Inputs”. This includes the AMBER MD input file “mdin”, the parameter and topology file “prmtop”, and 50 restart files “inpcrd” in subdirectories “prod1” to “prod50”. These 50 restart files were obtained by extracting coordinates at time intervals of 4 ns from a trajectory of 200 ns length followed by assignment of random velocities and short heating. For convenience, the geometries are also included as pdb file “inpcrd.pdb”. The “mdin” file is for a simulation with a time step of 2.0 fs using SHAKE constraints for bonds to hydrogen atoms at 300 K using the Berendsen thermostat with a time constant $\tau_T = 10\text{ps}$ for the heat bath coupling.

7 Validation: Structural Properties: Radius of gyration plots

Plots showing the radius of gyration R_g of ubiquitin for the 50 trajectories obtained with the CPU implementation and the different precision models (DPDP, SPDP and SPSP) of the PMEMD GPU implementation are shown in Figure 4.

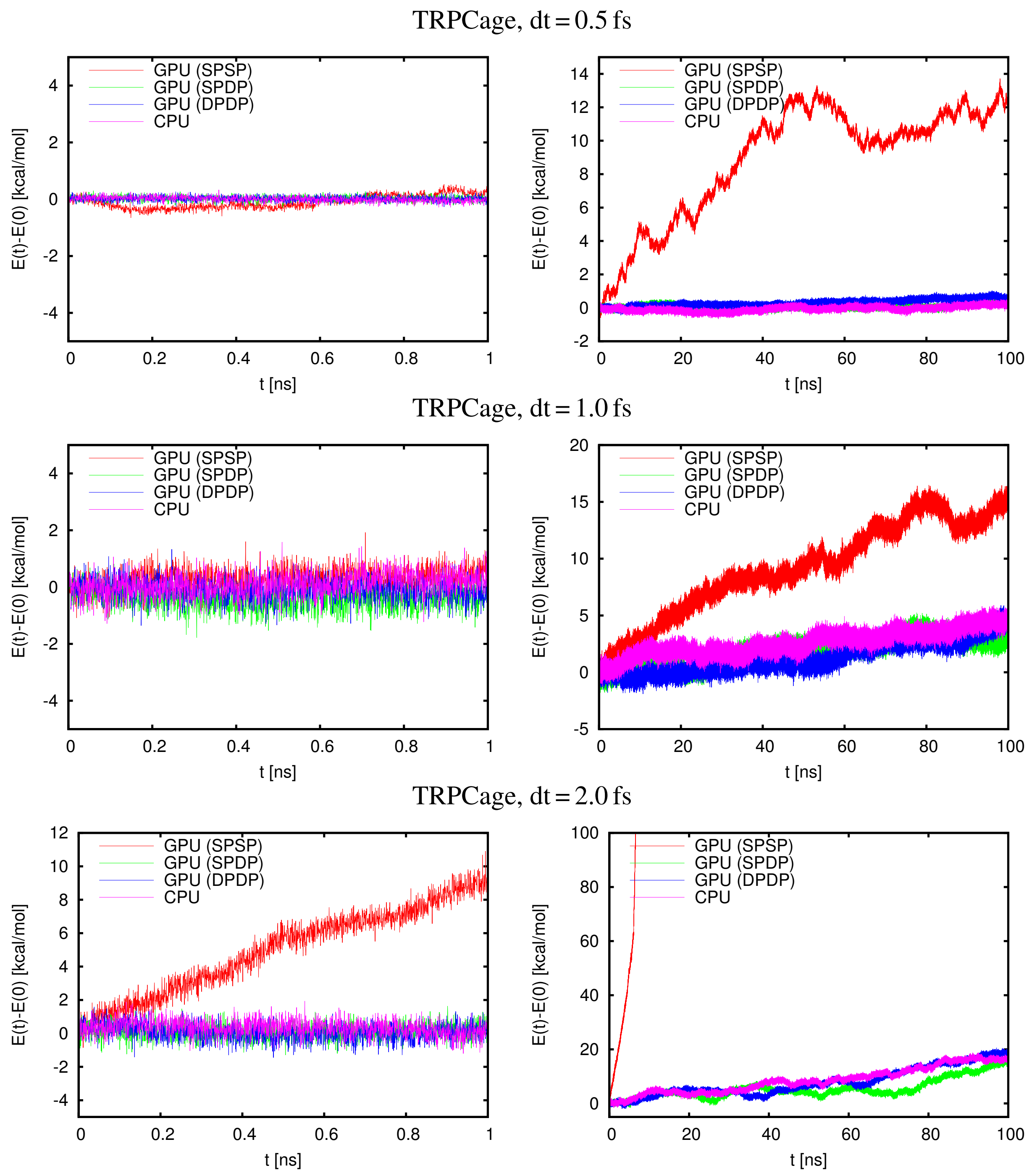

Figure 1: Total energy (kcal/mol) for different precision models along constant energy trajectories for TRPCage. Shown are results for a time step of 0.5 fs without constraints (top), a time step of 1.0 fs without constraints (middle) and a time step of 2.0 fs with constraints for bonds to hydrogen using the SHAKE algorithm (bottom). The left column shows the first nanosecond of each run while the right column shows the complete trajectory.

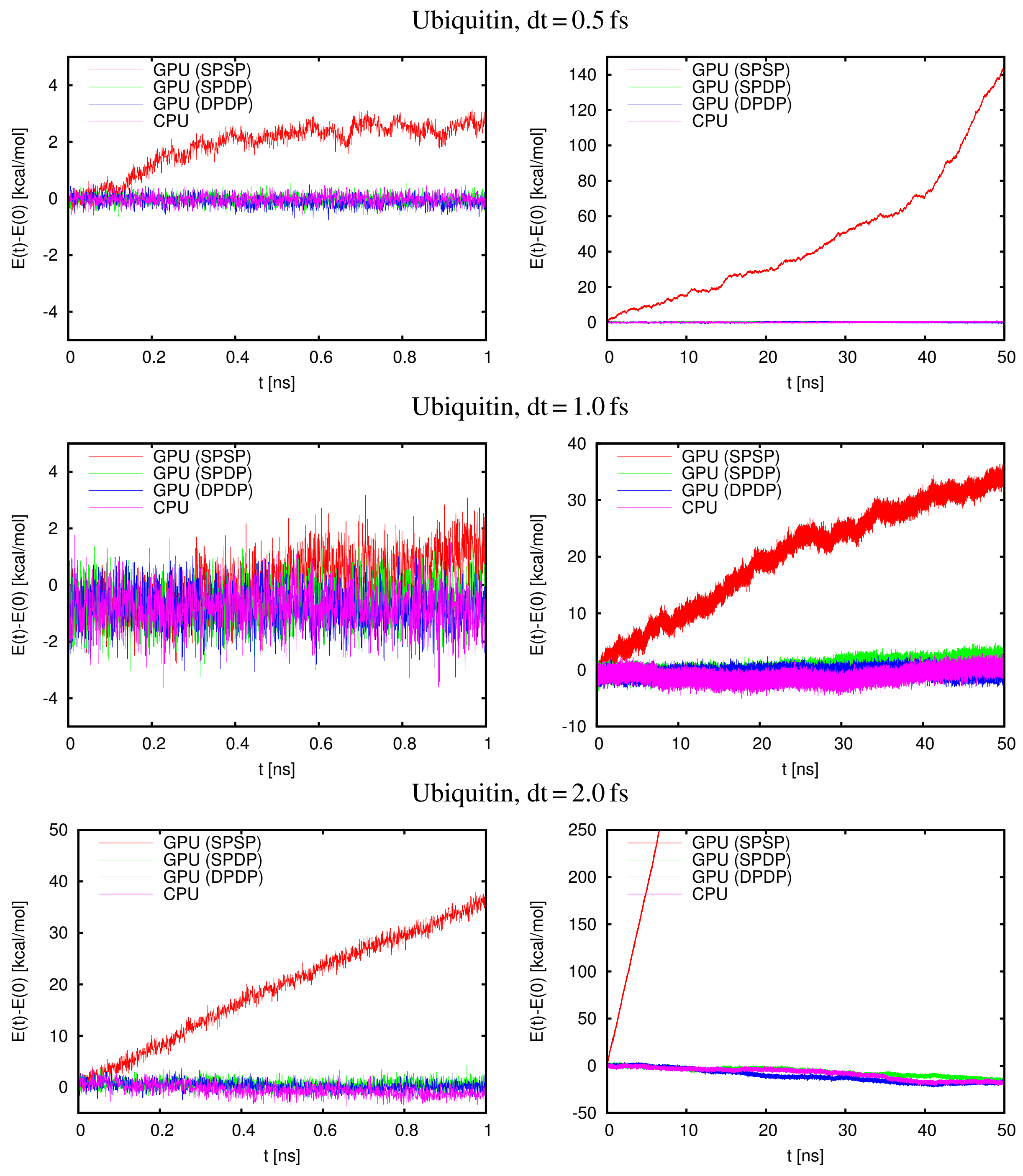

Figure 2: Total energy (kcal/mol) for different precision models along constant energy trajectories for ubiquitin. Shown are results for a time step of 0.5 fs without constraints (top), a time step of 1.0 fs without constraints (middle) and a time step of 2.0 fs with constraints for bonds to hydrogen using the SHAKE algorithm (bottom). The left column shows the first nanosecond of each run while the right column shows the complete trajectory.

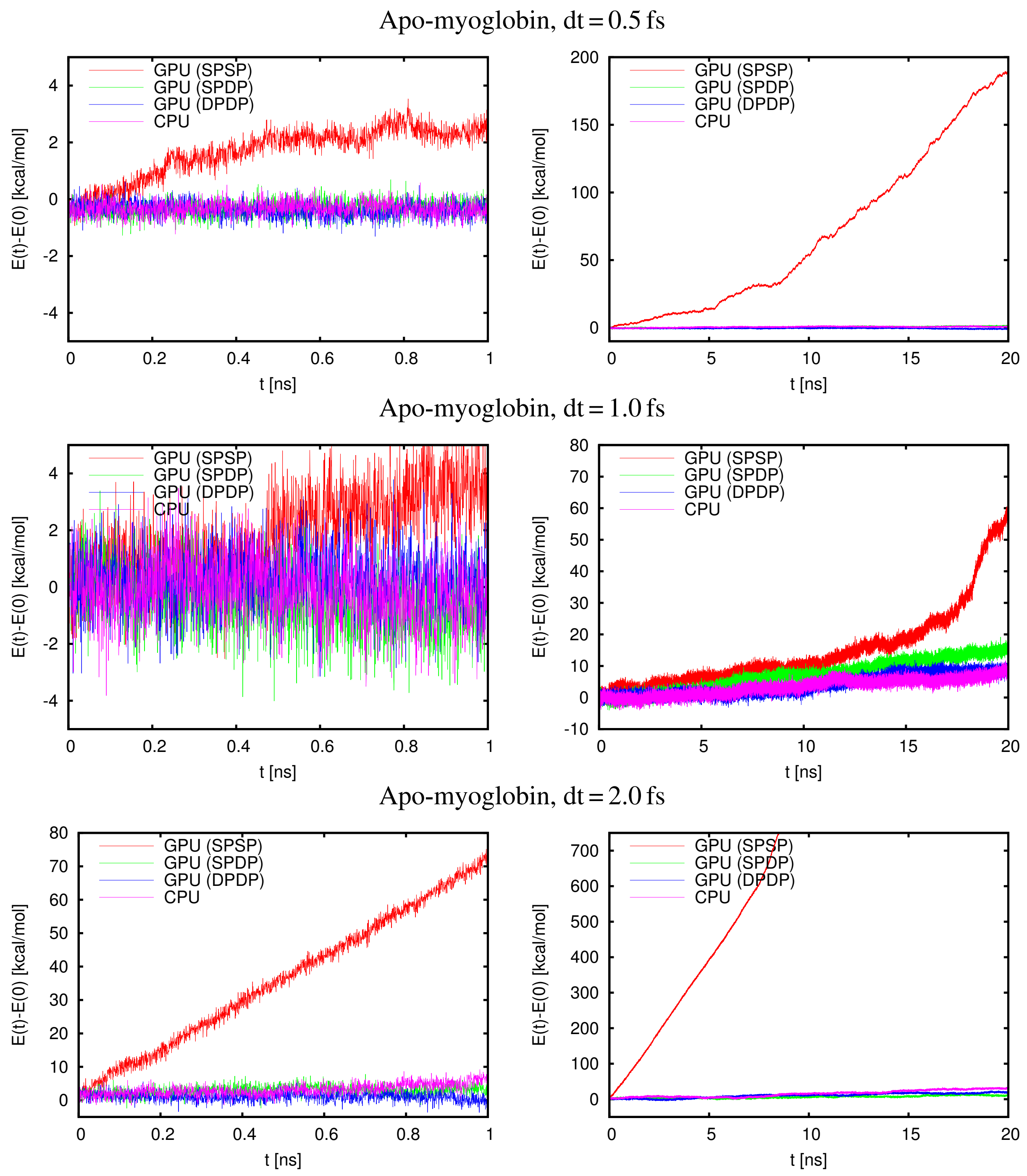

Figure 3: Total energy (kcal/mol) for different precision models along constant energy trajectories for apo-Myoglobin. Shown are results for a time step of 0.5 fs without constraints (top), a time step of 1.0 fs without constraints (middle) and a time step of 2.0 fs with constraints for bonds to hydrogen using the SHAKE algorithm (bottom). The left column shows the first nanosecond of each run while the right column shows the complete trajectory.

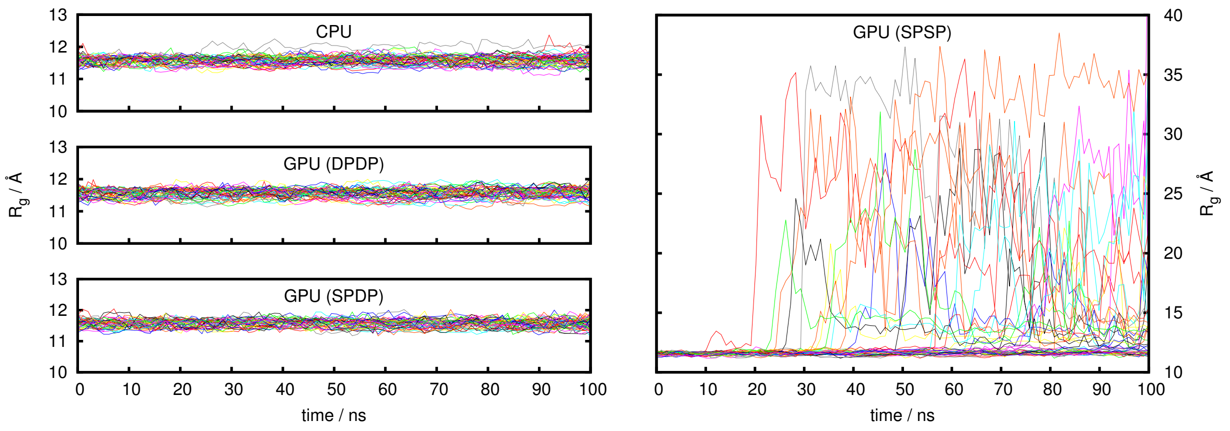

Figure 4: Radius of gyration R_g of ubiquitin for 50 independent trajectories of 100 ns length as obtained with the CPU implementation and the GPU implementation of PMEMD using different precision models.