PROTEINS
STRUCTURE ■ FUNCTION ■ BIOINFORMATICS

SOP-GPU: Accelerating biomolecular simulations in the centisecond timescale using graphics processors

A. Zhmurov,[1,2] R. I. Dima,[3] Y. Kholodov,[2] and V. Barsegov[1,2]*

[1] Department of Chemistry, University of Massachusetts, Lowell, Massachusetts 01854
[2] Moscow Institute of Physics and Technology, Moscow Region, Russia, 141700
[3] Department of Chemistry, University of Cincinnati, Cincinnati, Ohio 45221

ABSTRACT

Theoretical exploration of fundamental biological processes involving the forced unraveling of multimeric proteins, the sliding motion in protein fibers and the mechanical deformation of biomolecular assemblies under physiological force loads is challenging even for distributed computing systems. Using a C_α-based coarse-grained self organized polymer (SOP) model, we implemented the Langevin simulations of proteins on graphics processing units (SOP-GPU program). We assessed the computational performance of an end-to-end application of the program, where all the steps of the algorithm are running on a GPU, by profiling the simulation time and memory usage for a number of test systems. The ~90-fold computational speedup on a GPU, compared with an optimized central processing unit program, enabled us to follow the dynamics in the centisecond timescale, and to obtain the force-extension profiles using experimental pulling speeds (v_f = 1–10 μm/s) employed in atomic force microscopy and in optical tweezers-based dynamic force spectroscopy. We found that the mechanical molecular response critically depends on the conditions of force application and that the kinetics and pathways for unfolding change drastically even upon a modest 10-fold increase in v_f. This implies that, to resolve accurately the free energy landscape and to relate the results of single-molecule experiments *in vitro* and *in silico*, molecular simulations should be carried out under the experimentally relevant force loads. This can be accomplished in reasonable wall-clock time for biomolecules of size as large as 10^5 residues using the SOP-GPU package.

Proteins 2010; 78:2984–2999.

Key words: graphics processing units; Langevin dynamics; coarse-grained models; biomolecular assemblies; force-extension curves; dynamic force spectroscopy.

INTRODUCTION

Mechanical functions of protein fibers such as fibronectin, fibrin fibers, microtubules, and actin filaments are important in cytoskeletal support and cell motility,[1–3] in cell adhesion and the formation of the extracellular matrix,[4–7] and in blood clotting.[8–10] Physical properties of viral capsids of plant and animal viruses,[11–13] retroviruses,[14] and bacteriophages,[15,16] and the transitions between their stable and unstable states determine the life cycle of many viruses, including virus maturation, and infection of cells.[17] Single-molecule techniques, such as atomic force microscopy (AFM) and laser tweezer-based force spectroscopy, utilize either a time-dependent mechanical force $f(t)$ (force-ramp) or a constant force $f = f_0$ (force-clamp) to induce conformational transitions in biomolecules. These pioneering experiments have been used to study the physical properties of protein fibers[18–21] and viral capsids.[15,16,22,23] Yet, these experiments yield results that are nearly impossible to interpret without first having some a priori information about their free energy landscape.[9]

All-atom molecular dynamics (MD) simulations are widely used to access the submolecular behavior of biomolecules.[24–26] However, because all-atomic modeling is currently limited to a 10–50 nm length scale and 0.1–10 μs duration,[27–29] these methods allow mostly for the theoretical exploration of equilibrium properties of biomolecules, while reaching the biologically important ms-s timescale becomes virtually impossible even for a small system. In addition, to fully explore the free energy landscape underlying a biological process of interest, one needs to generate a statistically representative set of trajectories. One possibility is to carry out MD simulations on many-core computer clusters, but it requires tremendous computational resources and long central processing unit (CPU) times. For example, it takes 800000 CPU hours to obtain 20 short 1 ns MD trajectories for the southern

Additional Supporting Information may be found in the online version of this article.
Grant sponsor: American Chemical Society Petroleum Research Fund; Grant number: PRF #47624-G6; Grant sponsor: Russian Foundation for Basic Research; Grant number: #09-0712132; Grant sponsor: National Science Foundation; Grant number: MCB-0845002
*Correspondence to: V. Barsegov, Department of Chemistry, University of Massachusetts, Lowell, MA 01854. E-mail: valeri_barsegov@uml.edu.
Received 3 May 2010; Revised 3 June 2010; Accepted 2 July 2010
Published online 23 July 2010 in Wiley Online Library (wileyonlinelibrary.com).
DOI: 10.1002/prot.22824

bean mosaic virus, which contains as many as 4.5 million atoms, on an SGI Altix 4700 cluster.[30] These limitations exclude computations as an investigative tool in the study of a range of biological problems, such as the large deformations of protein fibers, the formation of biomolecular complexes and aggregates, and the mechanical failure of viral capsids, for which experimental data are already available, thereby rendering the direct comparison of the results of experiments *in vitro* and *in silico* impossible.

Graphics processors have evolved over the last few years into highly parallel, multithreaded computing devices. Unlike mainstream processor architecture, graphics processing units (GPUs) devote the majority of their logic units to performing actual calculations, rather than to cache memory and flow control. Massive multithreading, fast context switching, and high memory bandwidth enable GPUs to tolerate latencies associated with memory calls and to run many computational cores simultaneously. Programming tools for modern GPUs include several platforms such as ATI Stream Computing,[31] NVIDIA Compute Unified Device Architecture (CUDA),[32,33] and Open Computing Language (OpenCL).[34] Recent technological advances in the throughput-oriented hardware architecture of GPUs with extremely high peak arithmetic performance have unleashed tremendous computational power that has been utilized in a wide range of scientific applications, including calculations of electronic structure, *ab initio* quantum chemistry calculations, and quantum Monte Carlo simulations among many others.[35–39]

There exist preliminary versions of standard packages for MD simulations of proteins implemented on a GPU, such as NAMD,[28,40,41] Gromacs,[42] ACEMD,[43,44] and other applications.[45–47] However, due to inherent limitations in the scales of length and time, all-atom MD simulation methods, including those that are implemented on GPUs, cannot be applied to study the global unfolding transitions in biomolecules using experimental force loads $f(t) = r_f t$, where $r_f = \kappa v_f$ is the force-loading rate, and κ and v_f are the cantilever spring constant and the pulling speed, respectively. For example, in steered MD (SMD) simulations of proteins pulling speeds are 10^6–10^7-times faster than their experimental values ($v_f \approx$ 1–10 μm/s).[27,30] Under these far-from-equilibrium conditions, the kinetics, molecular mechanisms, and unfolding pathways observed in the pulling simulations at $v_f \approx 10^6$–10^8 μm/s and in the dynamic force measurements at $v_f \approx$ 1–10 μm/s might differ. At the theory level, one possibility is to adopt a coarse-grained description based on simplified models of proteins and Langevin Dynamics simulations.[48–50] Indeed, topology and the overall structure (geometry), rather than the atomic details, govern the force-driven unfolding of multimeric proteins and the large-scale mechanical deformation of protein fibers and viral capsids.[51]

We have developed and tested the GPU-based implementation of Langevin simulations using a C_α-based self organized polymer (SOP) coarse-grained model of the protein chain.[52] Previous studies have shown that the SOP model describes well the mechanical properties of proteins, including the Green Fluorescent Protein,[53] the tubulin dimer,[54] and kinesin.[55] The SOP-model has also been employed to explore the kinetics and to map the free energy landscape of various biomolecules such as the tetrahymena ribozyme,[52] riboswitch aptamers,[56] GroEL,[57] DHFR,[58] protein kinase A,[59] and myosin V.[60] Here we show that the combination of the SOP model and GPU-based computations (SOP-GPU program) enables one to carry out dynamic force measurements for a range of proteins *in silico* using experimental pulling speeds, to compare directly the experimental data with the simulation output, and, thus, to interpret the experimental results. To develop the SOP-GPU program, we used CUDA, a parallel computing environment (a dialect of the C and C++ programming languages) that provides a high level software platform for general purpose scientific applications. In what follows, we report on our implementation of the SOP-GPU package on the NVIDIA graphics card GeForce GTX 295.

First, we describe the methodology behind the GPU-based Langevin simulations, including the numerical routines for generating pseudorandom numbers, constructing Verlet lists, and integrating forward Langevin equations of motion. Second, we compare the results of CPU- and GPU-based simulations of the mechanical unfolding for a test system, the all-β-strand *WW* domain, and we assess the accuracy of the numerical integration. Next, we carry out pulling simulations for the *C2A* domain from human synaptotagmin (*Syt*1), and for the γ*C* chain and the double-*D* fragment from human fibrinogen (Fb). We compare directly the force-extension profiles, obtained using the time-dependent force protocol and pulling speeds v_f = 2.5 and 25 μm/s, with the experimental force spectra, and with the force-extension profiles generated at 10^4–10^6-times faster pulling speeds used in all-atom MD simulations. We show that the molecular mechanism(s), the kinetics, and thermodynamics of the biomechanical unfolding reactions all change with the pulling speed v_f. We also discuss the results of SOP-GPU simulations in terms of the simulation time, memory usage, and the computational speedup (i.e., CPU time vs. GPU time) for small proteins ($\lesssim$1000 residues), for fibrin fibers ($\lesssim$4000 residues), and for a large-size protein assembly—viral capsid *HK*97 (115,000 residues).

METHODS

GPU-based computations

The generally used CPUs have most of their logical elements dedicated to cache and flow control to employ complex logic and to provide computational cores with fast memory access. This makes a CPU capable of per-

forming computations following a sequential workflow. On a GPU, a large number of logical elements are devoted to actual computations, while cache and flow control are reduced to a small unit. These differences enable GPUs to achieve high arithmetic unit density and to perform the same computational procedure(s) simultaneously for all particles in a system. This is done by using many independent threads of execution running parallel calculations on different data sets.[61] For example, on a CPU, vector addition in M dimensions is performed in a loop, where the components of the resulting vector are computed one after another. On a GPU, this procedure can be performed using M independent threads, each of which computes just one component. Hence, on a GPU, a vector sum is computed at a cost of one addition, whereas on a CPU this cost is multiplied by M. Unlike computer clusters, where each core is capable of following its own computational protocol, most GPUs are based on the SIMD architecture (Single Instruction Multiple Data), which mandates that an identical instruction stream be fed to an array of processing units.[61]

To achieve top performance on a GPU, the numerical algorithms must be recast into a data-parallel form so that computational threads run the same instruction stream, but on different data. In addition, the task should be compute-intensive so that, most of the time, the GPU performs computations rather than reading and writing data. Because GPUs differ from CPUs in several fundamental ways, CPU-based methods for molecular simulations of biomolecules cannot be easily translated nor simply adapted to a GPU. Key architectural traits of modern GPUs and CPUs are compared in the Supporting Information (Section I). Yet, in molecular simulations, the particle-particle interactions are described by the same empirical potential energy function (force field), and the dynamics is obtained by solving numerically the same equations of motion for all particles. Hence, there is a direct correspondence between the SIMD architecture of a GPU at the hardware level and the numerical routines (software) used to follow the dynamics. It is then possible to execute a "single instruction", that is, the calculation of the potential energy, or the evaluation of forces, or the generation of random forces, or the integration of the equations of motion, on "multiple data" sets (many particles) at the same time using several arithmetic logic units (ALUs). This makes molecular simulations a natural candidate for implementation on a GPU.

Langevin dynamics simulations on a GPU

In this section, we describe the particle based and the interacting pair based methods for the parallel computation of potentials and forces due to binary (particle-particle) interactions. We outline the numerical procedures for the generation of Verlet lists and random forces, and for the numerical integration of Langevin equations of motion. A detailed description of the numerical algorithms can be found in Supporting Information (Section II). The algorithm decomposition, along with the workload division between the GPU device and the CPU, is diagrammed in Supporting Information Figure 1, where we displayed the computational workflow on the CPU and on the GPU, including operations on the data files for the molecular topology and for the particle energies and coordinates, and the data flow between the CPU DRAM and the GPU global memory (host-GPU data transfer).

There are two main optimization strategies that allow one to compute forces due to binary interactions. In the first approach, all forces acting on one particle are computed in one thread, which requires running N threads

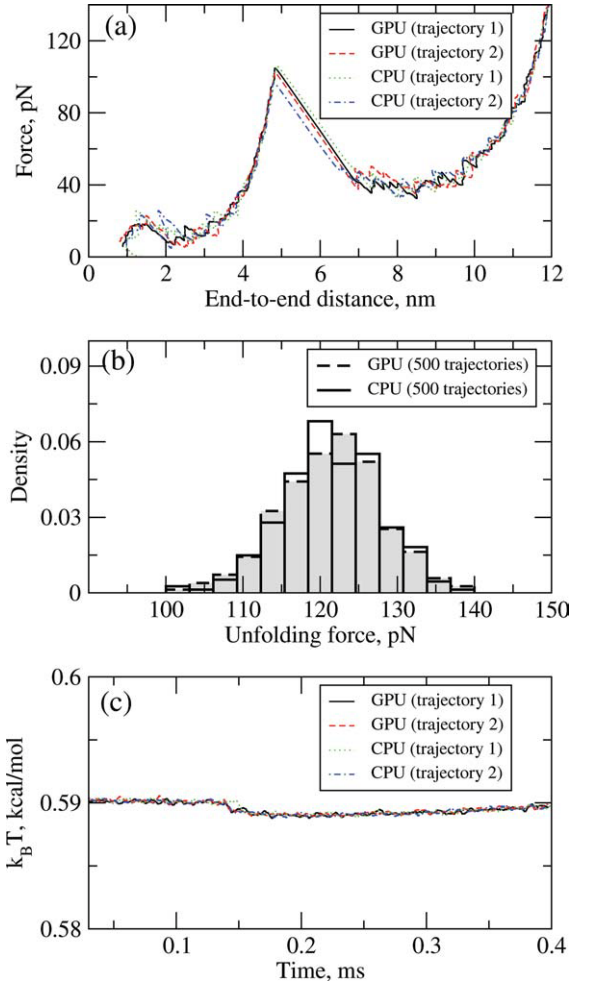

Figure 1

Comparison of the results of pulling simulations for the *WW*-domain (Table I) obtained on a GPU and on a CPU using the pulling speed $v_f = 2.5$ μ m/s. Panel **a**: Representative examples of the force spectrum (force-extension curves). Panel **b**: The histograms of unfolding forces, that is, the peak forces extracted from the force-extension curves, constructed using the bin size of $h_f \approx 3.1$ pN. Panel **c**: The time dependence of the average temperature of the protein chain $\langle k_B T(t) \rangle$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

to obtain the force values for all particles. We refer to this procedure as the particle based parallelization approach. The use of this approach requires the computation of the same force increment $\vec{df}$, acting on the ith and jth particles, twice in the ith and jth threads (Supporting Information, Section II).[47] In the interacting pair based parallelization approach, force calculations are performed for all (i,j)-pairs using P independent threads, which equals the number of pairs of interacting particles. Then, $2P$ force values are saved to different locations in the GPU global memory. We pursued both optimization strategies to exploit the data-parallel aspects of GPU-based computing (Supporting Information, Section II).

The particle based parallelization approach

In this approach, described by Anderson et al.,[47] N independent threads run on a GPU concurrently, each computing all the pair potentials for each particle and summing all the force values (except for the random force) to obtain the total force. Although the force exerted on the i-th and j-th particles is computed twice, the number of global memory calls is reduced by a factor of $2N$, and the time spent on recalculating the same potential is compensated by the time saved by not waiting to write/read the force data to/from the GPU global memory (Supporting Information, Algorithm 2). In Langevin simulations employing a native state-centric approach, the information about the covalent bonds and the native interactions, obtained from the PDB (Protein Data Bank[62]) structure of a protein, does not change. However, the information about nonbonded (long-range) interactions, describing the gradual attraction and the hardcore repulsion between pairs of residues, needs to be updated from time to time. The description of long-range interactions is the most computationally demanding component of the algorithm. A common approach is to take advantage of the fact that long-range interactions vanish over some distance. This allows one to use pair lists that include residues that are closer than some cutoff distance (Verlet lists).[63] Using the particle based parallelization approach, Verlet lists can be easily regenerated on a GPU to accelerate the computation of the potential energy function (Supporting Information, Algorithm 3).

The interacting pair based parallelization approach

To avoid computing the two-body potentials on a GPU twice, one can design a different algorithm, where each computational thread calculates a single pair potential for only two coupled residues. Forces exerted on the interacting residues (in opposite directions) are computed only once and all the forces exerted on each particle are summed up to obtain the total force. Although this approach requires additional memory calls, it enables one to accelerate the simulations when the number of residues N is close to the number of ALUs on the GPU, and/or when the calculation of pair potentials is computationally expensive. In this approach, each thread computes the forces for just one pair of residues, and the total number of threads equals the number of interacting pairs for one potential energy term (Supporting Information, Algorithm 4). This allows one to avoid memory conflicts, but requires an additional gathering kernel for the force summation (Supporting Information, Algorithm 5). On a GPU with Fermi architecture, thread safe addition of the force values helps remove the performance barriers associated with multiple memory calls.

In the interacting pair based parallelization approach, generating a Verlet list amounts to forming a vector of all the pairs of coupled residues for one potential energy term. On a GPU, constructing this vector is a formidable task, since the exact position in the list, to which the information about the next residue pair should be saved, is not known. One possibility is to use the atomicAdd(...) routine from C for CUDA,[32] which allows for integer addition in the GPU global memory without running into memory conflicts even when many threads attempt to access the same memory address at the same time. However, when threads work in parallel, identifying new pairs and saving them one after another may result in a Verlet list that is not ordered according to the particle index. This might lead to numerous noncoalescent memory reads and in an inefficient utilization of the cache memory. Thus, to obtain an ordered Verlet list, the list has to be sorted or updated on a CPU. Our estimates show that it is more efficient to compute interparticle distances on a GPU, to copy them to the CPU DRAM, and then to generate a new list.

Generation of random forces

Langevin simulations require a reliable source of normally distributed (pseudo)random numbers, $g_{i,\alpha}$ ($\alpha = x$, y, and z), to compute the three components of the Gaussian random force $G_{i,\alpha} = g_{i,\alpha}\sqrt{2k_BT\xi/h}$, where ξ is the friction coefficient and h is the integration time step. A random number generator (RNG) produces a sequence of random numbers $u_{i,\alpha}$ uniformly distributed in the unit interval [0,1], which is then translated into a sequence of normally distributed random numbers with zero mean and unit variance ($g_{i,\alpha}$) using the Box-Mueller transformation.[64] While there exist stand alone implementations of good quality RNGs on a GPU,[65] in Langevin simulations an RNG should be incorporated into the integration kernel to minimize the number of read/write calls. To produce random numbers during the numerical integration of Langevin equations, one can initiate an independent generator in each thread (one-

RNG-per-thread approach). First, a CPU generates N independent sets of random seeds for N RNGs, and then transfers them to the GPU global memory. When $4N$ random numbers $u_{i,\alpha}$ are needed for $3N$ normally distributed random numbers $g_{i,\alpha}$, each thread reads a corresponding set of random seeds to produce 4 normally distributed random numbers for each residue. Then, an RNG updates its current state in the GPU global memory, which is used as an initial seed within the same thread at the next time step. In a different approach, one RNG state can be shared among the computational threads across the entire GPU device (one-PRNG-for-all-threads approach). Using both strategies, we have developed and tested the GPU-based realizations of RNGs using the Hybrid Taus, Ran2, Lagged Fibonacci and Mersenne Twister algorithms (manuscript in preparation). All these algorithms pass stringent statistical tests and produce random numbers of very high statistical quality.[65] Here we employ the Hybrid Taus generator[65] (Supporting Information, Algorithm 6).

Numerical integration kernel

When the particle based parallelization is utilized, the subroutines for the force computation can be incorporated into the integration kernel. Although this requires more shared and local memory, it is efficient in simulations of smaller systems, for which N is comparable with the number of ALUs on a GPU. This allows one to read the coordinate variables, stored in the GPU global memory, only once at the beginning of the computational procedure and to pass the data to other subroutines. For larger systems ($N \sim 10^3$), there is a trade-off between the number of memory reads and the overall GPU utilization. Since all the interactions are more or less local, texture cache can be used as well to access the coordinates in the GPU global memory. When the interacting pair based parallelization is employed, the force computations can be performed in a separate kernel. Then, the force summation (gathering kernel, Algorithm 5 in Supporting Information) can be done inside the integration kernel. The numerical routine for the propagation of Langevin equations of motion is given in Supporting Information (Algorithm 7).

The SOP model

To describe the molecular force field, we adapted the SOP model (SOP-GPU program).[52–55] In the SOP model (see Section IV in Supporting Information), each residue is described using a single interaction center (C_α-atom). The potential energy function of a protein conformation V, specified in terms of the coordinates $\{r\} = r_1, r_2, \ldots, r_N$, is given by

$$V = V_{\mathrm{FENE}} + V_{NB}^{ATT} + V_{NB}^{REP} = -\sum_{i=1}^{N-1} \frac{k}{2} R_0^2 \log\left(1 - \frac{\left(r_{i,i+1} - r_{i,i+1}^0\right)^2}{R_0^2}\right) + \sum_{i=1}^{N-3}\sum_{j=i+3}^{N} \varepsilon_n \left[\left(\frac{r_{ij}^0}{r_{ij}}\right)^{12} - 2\left(\frac{r_{ij}^0}{r_{ij}}\right)^{6}\right]\Delta_{ij} + \sum_{i=1}^{N-2} \varepsilon_r \left(\frac{\sigma}{r_{i,i+2}}\right)^6 + \sum_{i=1}^{N-3}\sum_{j=i+3}^{N} \varepsilon_r \left(\frac{\sigma}{r_{ij}}\right)^6 \left(1 - \Delta_{ij}\right). \quad (1)$$

In Eq. (1), the finite extensible nonlinear elastic (FENE) potential V_{FENE} describes the backbone chain connectivity. The distance between two neighboring residues, i and $i + 1$, is $r_{i,i+1}$, while $r^0_{i,i+1}$ is its value in the native (PDB) structure, and $R_0 = 2$Å is the tolerance in the change of a covalent bond [first term in Eq. (1)]. We used the Lennard-Jones potential (V_{NB}^{ATT}) to account for the noncovalent interactions that stabilize the native state [second term in Eq. (1)]. We assumed that, if the non-covalently linked residues i and j ($|i - j|>2$) are within the cutoff distance $R_C = 8$Å in the native state, then $\Delta_{ij} = 1$, and zero otherwise. The value of ε_n quantifies the strength of the non-bonded interactions. All the nonnative interactions in the V_{NB}^{REP} potential are treated as repulsive [fourth term in Eq. (1)]. An additional constraint is imposed on the bond angle formed by residues i, $i + 1$, and $i + 2$ by including the repulsive potential with parameters $\varepsilon_r = 1$ kcal/mol and $\sigma = 3.8$ Å, which determine the strength and the range of the repulsion. This is done to ensure the self-avoidance of the protein chain.

To characterize the biomechanical unfolding reactions for each protein in terms of kinetic pathways, we used a combination of techniques. These include the visual inspection of the force-extension curves and the subsequent structural analysis of microscopic conformational transitions associated with each force peak. We also used a more detailed identification of transient conformations and intermediate states by comparing the global RMSD values ($\Delta_{WT}(t)$) with the partial RMSD values ($\Delta_{\Omega}(t)$) as described in Ref. 53. These measures of structural similarity between the wild-type (WT) or native state and a transient conformation (Ω) were used to estimate the time of detachment of a secondary structure element from the rest of the molecule in question.[53]

RESULTS

SOP-GPU program for Langevin simulations of proteins

We employed the methodology for the GPU-based Langevin dynamics to develop a CUDA program for biomolecular simulations fully implemented on a GPU. We carried out test simulations of the mechanical unfolding for

Table I
The Total Number of Residues, Covalent Bonds, Native Contacts, and Residue Pairs for a Range of Proteins (*WW* Domain, *Ig*27 domain, *C2A* Domain, and the γ*C* Chain), for Long Protein Fibers (Fb Monomer and Dimer $(Fb)_2$), and for the Protein Assembly (Viral Capsid *HK*97)

Protein	*WW*[a]	*Ig*27[b]	*C2A*[c]	γ*C*[d]	*D-D*[e]	Fb[f]	$(Fb)_2$[g]	*HK*97[h]
PDB code	1PIN	1TIT	2R83	1M1J	1FZB	3GHG	3GHG	1FT1[i]
Residues	34	89	126	517	1062	1913	3849	115140
Covalent bonds	33	88	125	521	1072	1932	3839	114720
Native contacts	65	255	328	1770	3498	5709	12560	467904
Non-native pairs	463	3573	7422	131,101	558,833	1,821,212	7,389,077	16,178,028[j]
Integration step h (*ps*)[k]	20	40	40	40	40	40	40	40
Number of steps[l]	0.4×10^9	0.5×10^9	0.5×10^9	1×10^9	1.5×10^9	2×10^9	4×10^9	1×10^{9m}
Trajectory length, s[n]	0.008	0.02	0.02	0.04	0.06	0.12	0.16	0.04

Also shown are the integration time step h, the number of iterations of the Langevin dynamics algorithm, required to generate a force-extension curve (force-indentation curve) for a protein (viral capsid *HK*97) at the experimental pulling speed v_f = 2.5 μ m/s, and the full length of the trajectory.
[a]All-β-strand *WW*-domain.
[b]*Ig*27 domain of human titin.
[c]*C2A* domain from human synaptotagmin-1 *Syt*1.
[d]γ*C*-Chains from human fibrinogen Fb.
[e]Double-*D* fragment (*D-D* interface) from human fibrinogen Fb.
[f]Human fibrinogen monomer Fb.
[g]Human fibrinogen dimer $(Fb)_2$ built from two monomers and the *D-D* interface.
[h]*HK*97 is Head II viral capsid.
[i]PDB code for a structural unit; the full *HK*97 structure is in the Viper database (66).
[j]Based on a cut-off distance of 200 Å.
[k]Integration time step is computed as described in Section IV in Supporting Information.
[l]Number of iterations required to fully stretch a protein.
[m]Number of iterations required to generate a force-indentation curve for *HK*97 at v_f = 2.5 μm/s
[n]Time required to obtain a full force-extension (force-indentation) curve for a protein (for *HK*97).

the all-β-strand domain *WW* from the human *Pin*1 protein (Table I) using SOP-GPU program. The smallest known all-β *WW* domain is of particular interest to the field of protein folding and dynamics, as evidenced by the considerable efforts expended to characterize the biophysical and biochemical properties of this protein.[29,67,68]

Benchmark simulations

We consider the following principal sources of error: (1) precision issues arising from the difference in single precision (GPU) and double precision (CPU) IEEE floating point arithmetic, (2) possible read/write errors associated with the use of the GPU global memory (hardware), and (3) overall accuracy of the SOP-GPU program, that is, possible errors in the numerical routines (software). We report on the performance of the SOP-GPU package (in CUDA) on a GeForce GTX 295, and compare it against the performance of the optimized C code (SOP-CPU program) on a dual Quad Core Xeon 2.83 GHz of a similar level of technology. All CPU/GPU benchmarks are obtained on a single GPU and on a single CPU core.

The Langevin equations of motion for each residue $\mathbf{r}_i$ have been integrated numerically using the first-order integration scheme,[69]

$$\mathbf{r}_i(t+h) = \mathbf{r}_i(t) + \frac{\mathbf{f}(\mathbf{r}_i(t))h}{\xi} + \mathbf{G}_i(t), \qquad (2)$$

where $\mathbf{G}_i = \mathbf{g}_i\sqrt{2k_BT\xi/h}$ is the random force and $\mathbf{f}(\mathbf{r}_i) = -(\partial V(\mathbf{r}_i)/\partial\mathbf{r}_i)$ is the total force, due to the covalent and the noncovalent interactions [Eq. (1)], exerted on the i-th particle. Benchmark simulations of the mechanical unfolding of the *WW* domain have been carried out at room temperature (k_BT = 4.14 pNnm) with the time step h = 20 ps using the standard bulk water viscosity, which corresponds to the friction coefficient $\xi = 7.0\times10^5$ pN ps/nm. The unfolding trajectories have been generated by fixing the *N*-terminal end and by pulling the *C*-terminal end of the *WW* domain with the time-dependent mechanical force, $f(t) = r_f t$. The force has been applied in the direction of the end-to-end vector using the force-loading rate $r_f = \kappa v_f$, where κ = 35 pN/nm is the cantilever spring constant and v_f = 2.5 μm/s is the pulling speed. It took 47 h to produce 1000 trajectories each of length 0.008 s (at v_f = 2.5 μm/s) by integrating over 4×10^8 steps on the GPU (Table I). This leads to a wall-clock time of ∼170 s per trajectory on a GPU when compared with 3.5 h on a CPU.

We analyzed the results of CPU- and GPU-based computations by comparing the force-extension curves (force spectra) $f(\Delta X)$, the average temperature $\langle T\rangle$, and the distributions of unfolding forces (peak forces in the force spectra) obtained on the CPU and on the GPU (Fig. 1). The temperature conservation ($d\langle T\rangle/dt$), the mechanical work performed on the system ($w = \int_{X_0}^{X_{fin}} f(X)dX$), and the distribution of unfolding forces are rigorous metrics of accuracy of pulling simulations. Aside from small deviations due to the different initial conditions, the profiles of $f(\Delta X)$ and $\langle T\rangle$, obtained on the CPU and on the GPU agree very well (Fig. 1). A small drop in $\langle T\rangle$ is due to the onset of the unfolding transition in the *WW* domain, which occurs at $t \approx 0.15$ ms. The histograms of

unfolding forces result in almost identical values of the average unfolding force, that is, $\langle f \rangle \approx 120.6$ pN (on the CPU) versus $\langle f \rangle \approx 120.9$ pN (on the GPU), and in similar values of the standard deviations of $\sigma_f \approx 5.8$ pN (on the CPU) versus $\sigma_f \approx 5.9$ pN (on the GPU). The magnitude of the most probable unfolding force is within the 60–200 pN force range observed for the β-strand proteins.[70,71] We also analyzed the numerical accuracy of the first-order integration scheme [Eq. (2)] by estimating the magnitude of numerical error, which might add up over many billions of iterations (Section III, Supporting Information). We found that, in long pulling simulations on a GPU, single precision arithmetic and the Ermak-McCammon algorithm can be used to describe accurately the mechanical properties of a biomolecule.

Performance measurements

We have compared the overall performance of an end-to-end application of the SOP-GPU program with the heavily tuned CPU-based implementation of the SOP model (SOP-CPU program) in describing the Langevin dynamics of the *WW* domain at equilibrium (Table I). We profiled the computational performance of the SOP-GPU program as a function of the number of independent trajectories, *s*, running concurrently on the GPU (many-runs-per-GPU approach). Alternatively, we could have run one trajectory on the GPU but for a range of systems of different number of residues *N* (one-run-per-GPU approach). Hence, in the former approach, the total system size is $N_{tot} = Ns$. The results obtained (Fig. 2a) show that, for the *WW* domain ($N = 34$), the GPU accelerates computations starting from 3 independent runs (many-runs-per-GPU approach), which is equivalent to a single run for a system of $N_{tot} \approx 102$ residues (one-run-per-GPU approach). This is the so-called break-even point [Fig. 2(a)]. It is important to understand that for small systems ($N_{tot} \lesssim 10^2$ residues), there is insufficient parallelism to fully occupy the GPU resources until the break-even point is reached. While the simulation time on the CPU scales linearly with *s* (or with N_{tot}), the scaling on the GPU in this regime is sublinear (nearly constant) up to $N_{tot} \approx 10^4$ ($s \approx 300$). At this point, the GPU shows significant performance gains relative to the CPU reaching the maximum 80–90-fold speedup [Fig. 2(a)]. We ran the simulations long enough to converge the speedup ratio (10^6 steps of size $h = 20$ ps). Beyond this point, the GPU is fully subscribed and the execution time scales linearly with *s* (N_{tot}).

The quad Core Xeon CPU E5440, used in this work, has a peak performance of ~45.28 GFlops or ~11.32 GFlops per computational core. GeForce GTX 295 (with 2 GPUs) has a peak performance of ~1788 GFlops or ~894 GFlops per GPU. Comparing one computational core of the CPU with one GPU, we find that, for fully optimized implementations on the GPU and on the CPU, the theoretical speed-up factor, given by the ratio of 894 GFlops (GPU) over 11.32 GFlops (CPU), is roughly equal to 80. The ~90-fold speedup obtained for a system of ~10^5 particles is very close to the theoretical value, which shows that SOP-GPU program is fully optimized [Fig. 2(b)]. The CPU/GPU time exceeds the theoretical estimate due to major differences in hardware architecture of the CPU and of the GPU, and due to higher memory bandwith on the GPU device.[32] Additional acceleration can be achieved using intrinsic high-speed mathematical functions present on the GPU [Fig. 2(b)].

In general, the total number of threads $M_{th} = m_B B$, defined by the number of thread blocks m_B of size *B*, is roughly equal N_{tot}, that is, $M_{th} \approx N_{tot} = Ns$. Because it is difficult to predict which block size *B* will result in the

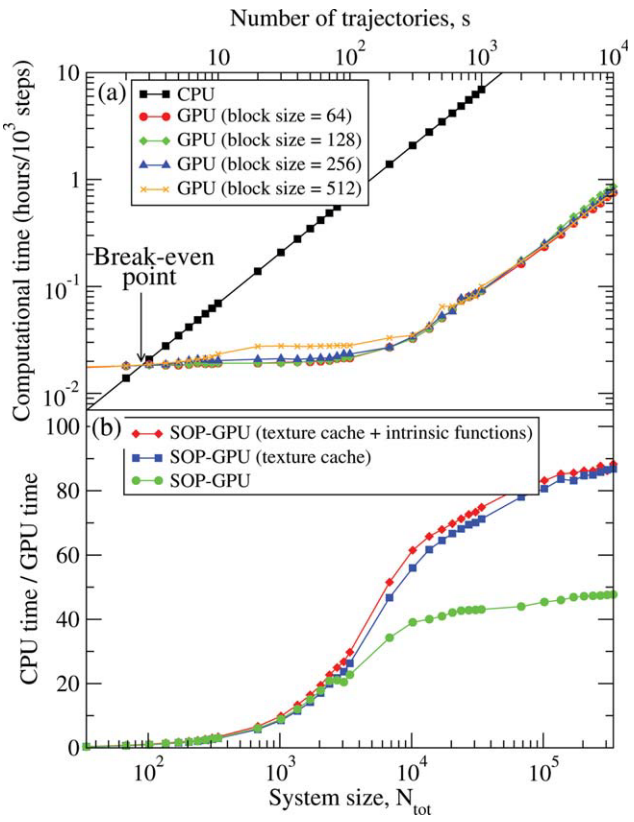

Figure 2

Panel **a**: The log-log plot of the computational time per 10^3 steps of the simulations on a CPU and on a GPU versus the system size, N_{tot} (one-run-per-GPU approach), and versus the number of independent trajectories running concurrently *s* (many-runs-per-GPU approach), for the all-β-strand *WW* domain. The relative GPU performance is tested for the blocks of size $B = 64$, 128, 256, and 512. While the single CPU core generates one trajectory at a time, the GPU device is capable of running many trajectories at the same time. Panel **b**: The log-linear plot of the relative CPU/GPU performance (computational speedup) of an end-to-end application of SOP-GPU program as a function of N_{tot} and *s*. The relative performance is compared for the SOP-GPU program, and for the SOP-GPU program accelerated by using texture cache, and by using texture cache plus intrinsic mathematical functions. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

best performance, we carried out benchmark simulations for $B = 64$, 128, 256 and 512. The results show that, to achieve top performance, M_{th} should exceed the number of ALUs on a GPU by a factor of 20–40. For example, on the graphics cards GeForce GTX 280, GTX 295, and Tesla C1060, each with 240 ALUs, $M_{th} \approx 5000$–10,000, which translates to $s \approx 300$ trajectories ($N_{\text{tot}} = 10{,}000$) in the case of the *WW* domain (Fig. 2). This implies that using small thread blocks is more advantageous when $M_{th} \lesssim 5000$, for example, when simulating $s = 1$ trajectory for a system of the size of the fibrinogen monomer Fb ($N \approx 2000$) or $s \approx 50$ runs for a system of $N \approx 100$ residues (*C2A*). Larger thread blocks should be used when $M_{th} \sim 5000$ to obtain a few trajectories for the fibrinogen dimer $(Fb)_2$ ($N \approx 4{,}000$) or one trajectory for the viral capsid *HK*97 ($\sim 10^5$ residues, Table I). We would like to point out that, in general, the block size B does not affect the total number of threads (M_{th}) on a GPU, and, hence, the ratio of M_{th} to the number of ALUs (for a given graphics processor). However, since one thread block is restricted to one multiprocessor, for a GPU to be fully subscribed the number of thread blocks $m_B = M_{th}/B$ should be larger than the number of multiprocessors. For example, on a graphics card GeForce GTX 295 the number of multiprocessors is 240/8 = 30, since there are 8 ALUs per multiprocessor (see Section I in Supporting Information).

Applications

Using SOP-GPU simulations, we carried out dynamic force measurements *in silico* for a range of proteins including the *C2A* domain from human synaptotagmin-1, and the γ*C*-chain and the double-*D* fragment from human fibrinogen (Table I). We employed the time-dependent protocol of force application, $f(t) = \kappa v_f t$ ($\kappa = 35$ pN/nm), using the experimental pulling speeds $v_f = 2.5$ and 25 μm/s (AFM). In all simulations at $T = 300$ K, the pulling force $f(t)$ was applied in the direction of the end-to-end vector of the protein chain. We compared the force-extension profiles and the unfolding pathways with the experimental results, and with the results of application of 10^4–10^6-times faster pulling speeds used in all-atom MD simulations (Supporting Information, Section IV).

The C2A domain from human synaptotagmin-1

Human synaptotagmin-1 (*Syt*1) is a Ca^{2+} sensor required for fast fusion of transmitter-loaded synaptic vesicles with the presynaptic plasma membrane.[72] The cytoplasmic region of *Syt*1 is composed of the *C2A* and *C2B* domains, which bind Ca^{2+} and induce monolayer bending. Hence, resolving the mechanical properties of *Syt*1 is crucial for understanding its precise role in synaptic fusion. AFM based dynamic force measurements have been conducted to probe the mechanical response of the *C2A* domain composed of eight β-strands (*S*–*S*8) and one α-helix (*H*1).[71,73] We explored the unfolding micromechanics of *C2A* using SOP-GPU simulations (Supporting Information, Section IV). On a single GPU, it takes 35 h to generate 100 long 0.02 s trajectories of unfolding (at $v_f = 2.5$ μm/s) over 5×10^8 steps using the many-runs-per-GPU approach (21 min per trajectory, Table I).

In the 130 trajectories of unfolding at $v_f = 2.5$ μm/s, the *C2A* domain was observed to unravel following either the major pathway (69% of the time), or the minor pathway (31%) displayed in Figure 3(a). These results agree well with the experimental data,[73] which showed that

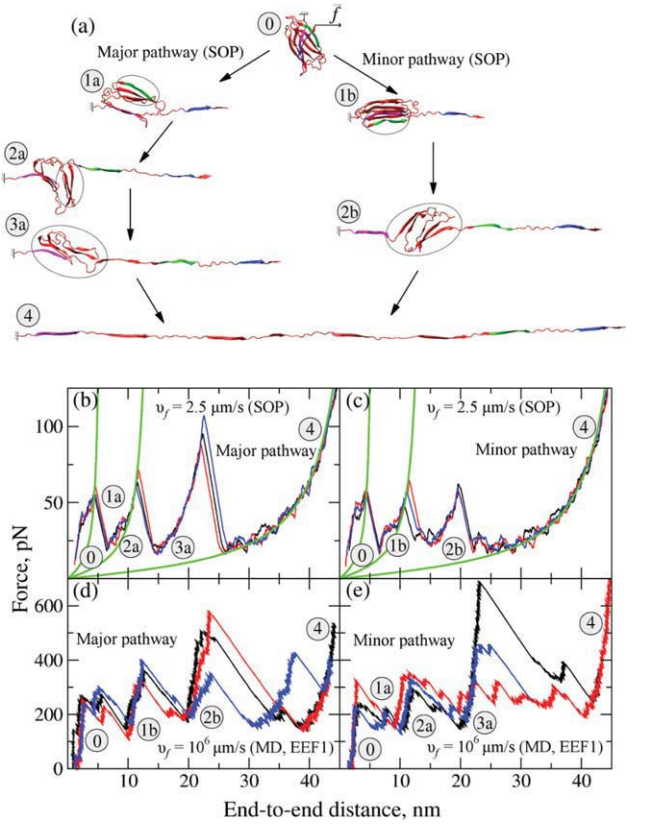

Figure 3

The dynamics of unfolding of the *C2A* domain from human synaptotagmin *Syt*1 (Table I) obtained using SOP-GPU simulations at the pulling speeds $v_f = 2.5$ and 25 μm/s (panels **a**, **b**, and **c**), and using the all-atom MD simulations in implicit solvent on a CPU at $v_f = 10^6$ μm/s (panels d–e). Panel **a**: Structural snapshots of the native state (0), the partially unfolded intermediate states (1a, 1b, 2a, 2b, and 3a), and the fully stretched state.[4] The intermediate structures 1a, 2a, and 3a (major pathway), and 1b and 2b (minor pathway) are observed both in the SOP-GPU simulations and in the MD simulations. However, the major unfolding pathway observed most of the time in the experiments and in the SOP-GPU simulations at $v_f = 2.5$ μm/s is the minor pathway observed in the MD simulations. Panels **b** and **c**: The force-extension profiles (rugged curves) obtained from the SOP-GPU simulations at $v_f = 2.5$ μm/s, which correspond to the major pathway (panel b) and to the minor pathway (panel c). The ascending curves are the worm-like chain fits to the force peaks. Panels **d** and **e**: The force-extension profiles obtained using MD simulations, which correspond to the major pathway (panel d) and to the minor pathway (panel e). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

~38% of the time *C2A* domains unfold by populating a partially unfolded intermediate. Interestingly, the percentages change with the pulling speed: in 64% (36%) of trajectories obtained at a 10-fold faster pulling speed v_f = 25 μm/s, the *C2A* domain was observed to follow the major (minor) pathway. Along both pathways, the first unfolding transition is due to the forced unraveling of the *C*-terminal strand *S*8 at a 58.5 ± 4.6 pN force [structures 1a and 1b in Fig. 3(a)], which compares well with the average experimental force of 51 ± 14 pN.[71,73] The next force peak at 61.4 ± 5.7 pN is due to unfolding transitions which differ in the major and in the minor pathways. Along the major pathway, the *C2A* domain unravels gradually from the *C*-terminal end through the detachment of the *S*7 strand [structure 2a, Fig. 3(a)] and the *S*6 strand [structure 3a, Fig. 3(a)]. Along the minor pathway, the second unfolding transition is due to the simultaneous unraveling of the strands *S*1 and *S*7 at both ends of the molecule [structure 2*b*, Fig. 3(a)]. The corresponding ~7.0 nm peak-to-peak distance [Fig. 3(b,c)] is close to 7.4 ± 3.5 nm extension measured experimentally [see Fig. 3(c) in Ref. 73].

We also carried out pulling simulations for the *C2A* domain using MD simulations in implicit solvent (see Supporting Information). It took 4 days to generate one trajectory of unfolding at ~10^5 faster pulling speed $v_f = 10^5$ μm/s on one dual-core 2.83 GHz AMD Opteron 2214 node. At this v_f, the *C2A* domain unravels following three pathways, two of which are observed at the slower speeds v_f = 2.5 and 25 μ m/s [major and minor pathways, Fig. 3(a)]. In the additional third pathway, the *C2A* domain unfolds starting from the *N*-terminal end of the chain (data not shown). We also found that a ~10^5-fold increase in v_f changes the pathway probabilities. The minor pathway, observed in the SOP-GPU simulations at v_f = 2.5 μ m/s, becomes the dominant pathway (55%) in all-atom MD simulations [Fig. 3(d)], whereas the role of the major pathway in the SOP-GPU simulations diminishes (27%) in MD simulations [Fig. 3(e)]. The remaining 18% of MD simulation runs correspond to the new (third) pathway. Hence, a ~10^5-fold increase in v_f triggers a substantial change in the unfolding pattern and in the pathway probabilities, and leads to the emergence of a new pathway, which is not observed at the experimental pulling speeds. We see that, although all-atom MD simulation methods do capture the complexity of the mechanical unfolding and show some quantitative agreement with the experimental data, their interpretive capacity is reduced due to lack of quantitative agreement with experiments in terms of the unfolding kinetics (pathways) and thermodynamics (force-extension curves).

The γ*C*-chain and double-*D* fragment from human fibrinogen

Fibrinogen (Fb) is a blood plasma protein that polymerizes into fibrin fibers, which aggregate to form the three-dimensional scaffold called a clot. A blood clot needs to maintain a balance between the stiffness and the plasticity required for hemostasis and wound healing. Although the physical properties of fibrin fibers have been characterized at the fiber and whole clot levels, the micromechanics of fibrinogen, the precursor of fibrin, remains unexplored. AFM based dynamic force measurements have been conducted to probe the mechanical response of fibrin oligomers.[9,21] We carried out SOP-GPU pulling simulations for the γ*C*-chain and for the double-*D* fragment of Fb (Table I). The double-*D* fragment is composed of the two identical subunits (*D*) formed by the γ*C*- and β*C*-chains of Fb, interacting through the *D*:*D* association interface.

On a single GPU, it takes ~98 h to generate 20 long (0.04 s) trajectories of unfolding for the γ*C*-chain (at v_f = 2.5 μm/s) over 10^9 steps using the many-runs-per-GPU approach (5 h per trajectory, Table I). The sawtooth force-extension profiles (Fig. 4) compare well with the experimental force spectra of Brown et al. (see Fig. 3 in Ref. 21), obtained at the same value of v_f. The unfolding forces and molecular elongations observed in our SOP-GPU simulations are within the experimental ~50–150 pN force range and in the ~15–40 nm extension range. The unfolding mechanism consists of three major steps, corresponding to the three force peaks in the force-extension curves (Fig. 4). First, the *C*-terminal β-strand (residues 381–390) is pulled out of the central part of the γ*C*-chain, which is followed by unfolding of the central portion of the chain (residues 234–311). This results in the unraveling of the γ*C*-chain into two subunits [structure 1 in Fig. 4(a)]. Next, the first subunit (residues 139–234) and the second subunit (residues 311–381) unfold one after another [structures 2a and 2b in Fig. 4(a)]. This mechanism persists as it appears in all 100 simulation runs at v_f = 2.5 μm/s. Because of the two disulfide bonds (*Cys*153–*Cys*183 and *Cys*326–*Cys*339), the globally unfolded state [structure 3 in Fig. 4(a)] shows the two remaining globular domains. We see that the force spectra obtained at the slower pulling speed v_f = 2.5 μm/s [Fig. 4(c)] show clear force signals, compared to the spectra generated at the faster speed v_f = 25 μm/s [Fig. 4(d)]. This implies that the unfolding transitions are resolved better at v_f = 2.5 μm/s. For comparison, we also carried out pulling simulations at ~10^3-times faster v_f of 10^3 μm/s. The resulting force spectra show a single broad force peak of ~300 pN corresponding to the end-to-end distance of ~10–20 nm [Fig. 4(c,d)]. At this pulling speed, the secondary structure motifs of the γ*C*-chain unravel almost simultaneously.

We also performed two all-atom SMD simulation runs for the γ*C*-chain (see Supporting Information). It took 37 days to obtain one trajectory on one node equipped with two quad-core 2.83 GHz Intel Xeon processors. At a 10^4-times faster pulling speed $v_f = 3 \times 10^4$ μm/s, the γ*C*-chain unravels in just one step at a much higher force of ~1250 pN [Fig. 4(e)], compared to the experi-

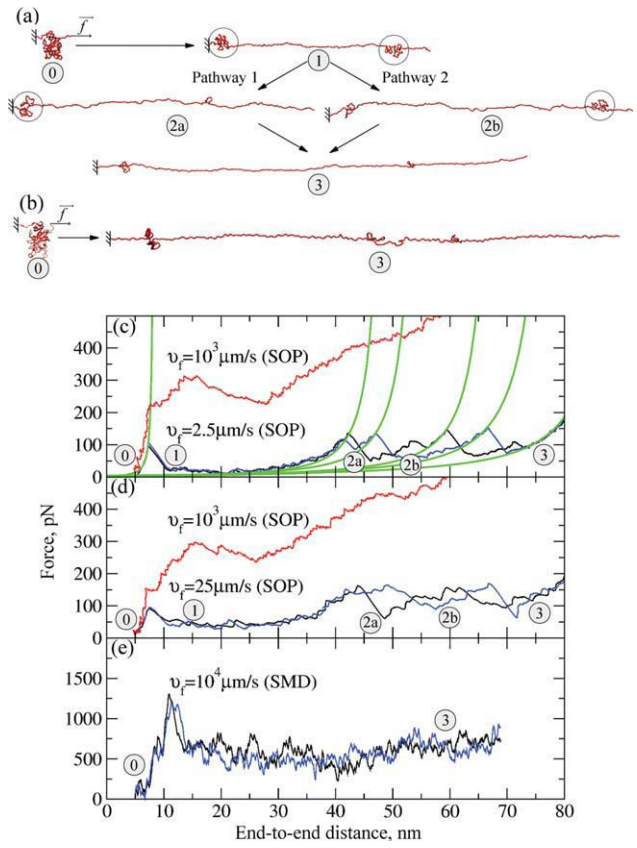

Figure 4

The dynamics of unfolding of the γ*C*-chain from human fibrinogen Fb (Table I) obtained using SOP-GPU simulations at the pulling speeds v_f = 2.5 and 25 μm/s (panels **a**, **c**, and **d**), and using SMD simulations at $v_f = 3 \times 10^4$ μm/s on a CPU (panels **b** and **e**). Panels **a** and **b**: Structural snapshots of the folded state (0), the intermediate partially unfolded states (1, 2a, and 2b), and the globally unfolded state.[3] The intermediate states 1 and 2a (for pathway 1), and 1 and 2b (for pathway 2) are observed only in SOP-GPU simulations, but not in SMD simulations. Panels **c–e**: The force-extension profiles (rugged curves) generated at v_f = 2.5 μm/s (pathways 1 and 2, panel **c**) and at v_f = 25 μm/s (pathways 1 and 2, panel **d**) using the SOP-GPU program, and at $v_f = 10^6$ μm/s utilizing SMD simulations (panel **e**). For comparison, we also present the force-extension curve obtained from two SOP-GPU simulation runs at $v_f = 10^3$ μm/s (panels **c** and **d**). The ascending curves are the worm-like chain fits to the force peaks. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

mental ~50–150 pN force range,[21] populating the globally unfolded state [structure 3 in Fig. 4(b)]. Similar to the results of the SOP-GPU pulling simulations at $v_f = 10^3$ μm/s, in the SMD simulations at $v_f = 3 \times 10^4$ μm/s the consecutive unfolding transitions observed at v_f = 2.5 and 25 μm/s occur simultaneously. The end-to-end distance of ~10–20 nm, compares well with the end-to-end distance observed in the SOP-GPU simulations at $v_f = 10^3$ μm/s. Hence, at a ~10^3–10^4-fold faster pulling speed, both the SOP-GPU and the SMD simulations predict a molecular mechanism for unfolding that is markedly different from the mechanism observed at the experimental pulling speeds. As a result, the rich structure of the force spectrum with multiple peaks, separated by the average peak-to-peak distance of ~15–40 nm, disappears altogether.

On a single GPU, it takes ~4 days to generate 50.06 s long, trajectories of unfolding for the double-*D* fragment (at v_f = 2.5 μm/s) over 1.5×10^9 steps using the many-runs-per-GPU approach (18 h per trajectory, Table I). We found that, at 2.5 μm/s, the unfolding process starts with the force-induced rupture of the binding contacts stabilizing the *D*:*D* interface. This results in a ~4–5 nm extension, and in the formation of mechanically decoupled identical *D* fragments [structure 1 in Fig. 5(a)]. Next, the γ*C*-chains in the *D* domains unfold following the three major steps observed for the single γ*C*-chain [Fig. 4(a)], leading to the formation of six interme-

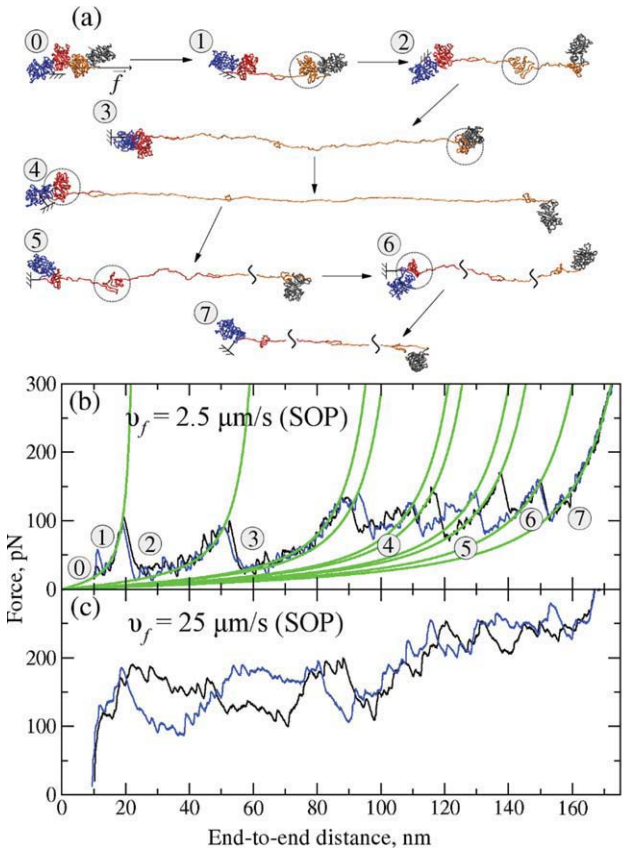

Figure 5

The dynamics of unfolding of the double-*D* fragment from human fibrinogen (Table I) obtained using SOP-GPU simulations at the pulling speed v_f = 2.5 μm/s (panels **a** and **b**) and 25 μm/s (panel **c**). Panel **a**: Structural snapshots of the native state (0), the intermediate conformation with the disrupted *D*:*D*-interface,[1] the partially unfolded intermediate states,[2–6] and the fully stretched state.[7] Panels **b** and **c**: The force-extension profiles (rugged curves) generated at v_f = 2.5 μm/s (panel **b**) and at v_f = 25 μm/s (panel **c**) using SOP-GPU program. Also shown are the worm-like chain fits to the force peaks. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

diate conformations [structures 2–7, Fig. 5(a)]. This mechanism is robust as it was observed in all 20 runs. The force spectra, obtained at the slower $v_f = 2.5$ μm/s, agree very well with the experimental force spectra (Fig. 3 in Ref. 21). At the faster pulling speed $v_f = 25$ μm/s, several unfolding transitions described above occur almost simultaneously. The shift in the unfolding mechanism is fully reflected both in the qualitative and in the quantitative changes to the force spectra [Fig. 5(c)], where some of the force peaks are not resolved. Indeed, the discrete unfolding steps observed at $v_f = 2.5$ μm/s, each resulting in a distinct force signal [Fig. 5(b)], disappear when the double-*D* fragment is pulled faster.

DISCUSSION

GPU-based Langevin simulations in the centisecond timescale

We have developed and tested, to the best of our knowledge, the first GPU-based implementation of Langevin simulations of biomolecules. We presented the numerical routines for the generation of random numbers (Hybrid Taus algorithm), for the construction of Verlet lists, and for the numerical integration (first order scheme). The algorithm has been mapped into a standard CUDA code (SOP-GPU program). Benchmark tests show that the results of simulations of the mechanical unfolding of proteins on a GPU and on a CPU agree very well (Fig. 1). This attests to the accuracy of the SOP-GPU program. Using an exactly solvable model of the Brownian particle evolving on a harmonic potential, we assessed the accuracy of the numerical integration of Langevin equations of motion. We showed that using the first-order integrator (Ermak-McCammon algorithm) is sufficient to describe accurately the force-induced elongation of a protein chain over many billions of iterations.

GPUs can be utilized to generate a few trajectories of Langevin dynamics for a large system of $\sim 10^3$–10^5 residues using the one-run-per-GPU approach. This is the main approach taken by many researchers in MD simulations on graphics processors.[28,42,43,46,47,74] One of the highlights of this work is the many-runs-per-GPU approach, which has never been benchmarked. This approach enables one to run many independent trajectories simultaneously for a small system of $\sim 10^2$–10^3 residues. Another important highlight of this work is an efficient utilization of the GPU texture cache to process intermediate variables and to store data locally. Indeed, compared with the optimized CPU-based implementation of the Langevin dynamics algorithm, the GPU-based realization without texture cache leads to a moderate ~10–50-fold computational speedup, which also depends on the system size $N_{tot} \approx 10^3$–10^6 [Fig. 2(b)]. However, efficient utilization of the texture memory to access the coordinates of the particles in the GPU global memory

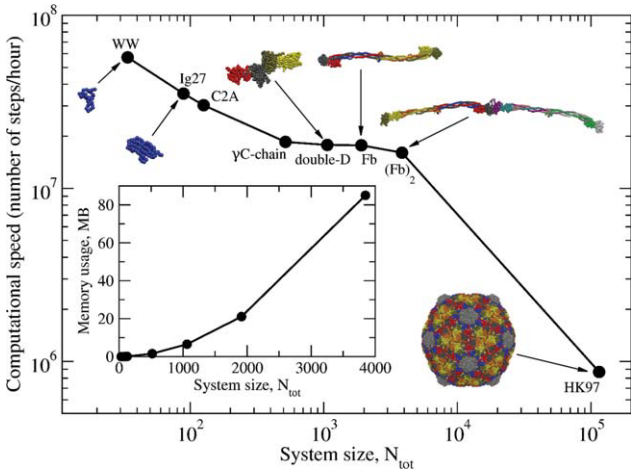

Figure 6

The log-log plot of the computational speed (number of steps per hour) for an end-to-end application of the SOP-GPU program (with thread blocks of size $B = 256$), accelerated using texture cache and intrinsic mathematical functions, versus system size N_{tot} (one-run-per-GPU approach). Shown are the estimates for a range of biomolecules, including small proteins (the *WW*-domain, the *Ig*27 domain, and the *C2A* domain), large protein fragments (the γ*C*-chain and the double-*D* fragment), long protein fibers (the Fb monomer and dimer $(Fb)_2$), and the large-size viral capsid *HK*97 (Table I). Inset: The memory usage on a GPU as a function of N_{tot}. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

allowed us to reach an 85-fold speedup. Additional acceleration can be achieved using intrinsic high-speed mathematical functions [90-fold speedup, Fig. 2(b)]. We profiled the SOP-GPU program in terms of the computational time and memory usage for a range of proteins listed in Table I. The simulation time on a GPU remains roughly constant for a system of $N_{tot} \approx 10^2$–10^4 residues (sub-linear regime), and scales linearly with $N_{tot} > 10^4$ residues.

We also ran benchmark simulations for several different protein systems using the one-run-per-GPU approach, and analyzed one simulation run for each system, generated over 10^8–10^9 steps. For all test systems of $N_{tot} \lesssim 4000$ residues, the computational speed exceeded 10^7 steps per wall-clock hour (Fig. 6). This is not surprising since $M_{th} \approx 5000$ is the amount of threads needed to fully utilize the GPU resources. For larger systems of $N_{tot} \gtrsim 10^4$ residues, the computational speed scales roughly linearly with N_{tot} (Figs. 2 and 6). The nonmonotonic dependence of the computational speed on N_{tot} is due to the different native topologies of the test systems, that is the number of native and non-native contacts in the PDB structure (Table I). We showed that the amount of GPU on-board memory in contemporary graphics cards, that is, ~1 GB (GeForce GTX 200 series) and 4 GB (Tesla C1060), is sufficient to describe the Langevin dynamics of large biomolecular systems of $\sim 10^4$ residues, that is, comparable in size with the fibrinogen dimer $(Fb)_2$ (inset in

Fig. 6). The amount of on-board memory in graphics cards with new Fermi architecture (from NVIDIA) is 1.5–6 GB.[75] We also described an algorithm, a combination of Verlet lists and pair lists for non-native pairs, for efficient use of the limited GPU global memory for larger systems (see Supporting Information, Section II), such as the viral capsid *HK*97 (N_{tot}~10^5).

A combination of the C_α-based coarse-grained SOP model of proteins and GPU-based computations enabled us to follow the biomolecular dynamics in the long 0.01–0.1 s timescale (Table I). This is the experimentally relevant time for biomechanical reactions, including the forced unfolding of small proteins and protein tandems,[20,53,73,76] the force-driven elongation of long protein fibers,[21] and the force-induced indentation of large-size biomolecular assemblies such as plant and animal viruses[11–14] and bacteriophages.[15,16] Hence, SOP-GPU simulations can be utilized to explore the unfolding micromechanics of protein fibers, and to characterize the visco-elastic properties of viral capsids using the experimental force loads. This makes it possible to interpret the experimental force spectra and force-indentation profiles of biomolecules, obtained in dynamic force spectroscopy assays, and, thus, to bridge the gap between theory and experiments. For example, on a single GPU (GeForce GTX 295) it takes only 10 days to generate a single trajectory of unfolding at v_f = 2.5 μm/s over 4 × 10^9 steps for the fibrinogen dimer $(Fb)_2$. For comparison, it would take ~16 months to complete one simulation run on the CPU with 2.83 *GHz* Intel Xeon E5440 processor (11,520 CPU hours). It takes ~40 days to generate one force-indentation curve (both indentation and retraction) for the viral capsid *HK*97 (at v_f = 2.5 μm/s) over 10^9 steps on a single GPU (GeForce GTX 200 series, Tesla C1060), compared with 120 months on a CPU with 2.83 *GHz* Intel Xeon E5440 processor (86,400 CPU hours). Hence, many force-indentation trajectories for a viral capsid can be obtained on a single desktop computer equipped with several GPUs. A typical force-indentation profile for the viral capsid *HK*97 obtained at v_f = 2.5 μm/s is presented in Figure 7, which shows the hysteresis upon retraction observed in AFM experiments on viral capsids.[16,17] The estimated spring constant of ≈0.12 N/m for *HK*97 agrees with the experimental values of this parameter for empty capsids.[15,77]

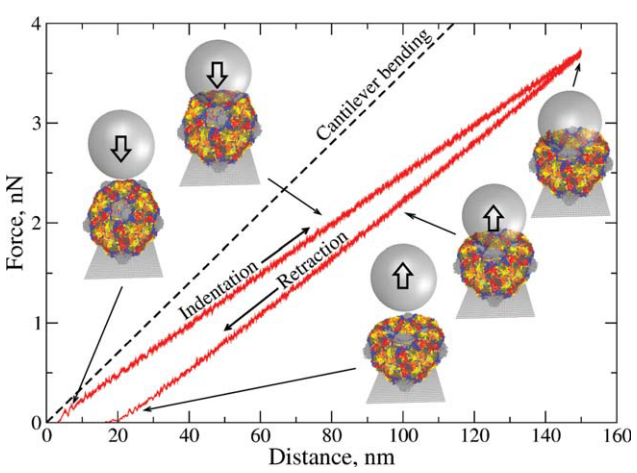

Figure 7

The dynamics of force-induced indentation of the empty viral capsid *HK*97 (Table I) obtained using SOP-GPU simulations at the pulling speed v_f = 2.5 μm/s. The spherical capsid is pressed against the surface by the cantilever tip. The direction of force application is indicated by the arrow. The ascending and descending curves showing the indentation and retraction profiles for *HK*97 are compared with the "control" simulation run describing the mechanical bending of the cantilever tip (dashed line). Also shown are the transient structures encountered along the indentation-retraction pathway.

Unfolding dynamics: slow versus fast pulling speeds

The nature of the mechanical response of soft biological matter depends sensitively on the conditions of force application $\mathbf{f} = f(t)\mathbf{n}$, including the force magnitude $f(t) = r_f t$ which is proportional to the force-loading rate $r_f = \kappa v_f$ (or the pulling speed v_f) and the direction $\mathbf{n}$ (e.g., shearing force vs. peeling force). In this picture, $f(t)$ projects the multidimensional free energy landscape of the protein of interest on the singled out one dimensional reaction coordinate, parametrized by the protein end-to-end vector $\mathbf{X}$. The applied force tilts the energy landscape, thus, rendering the native folded state unstable when $f(t)$ exceeds some critical threshold. On the other hand, the internal dynamics of the protein chain couples the local topology-dependent modes to the global unfolding transitions reflected in the time-evolution of $\mathbf{X}$.[78] As a result, the mechanical perturbation ramped up with v_f might be shared unequally among the secondary structure motifs due to their different mechanical stability and/or due to time-dependent tension propagation from the tagged *C*-terminus to the constrained *N*-terminus of the molecule. The influence of tension propagation on the unfolding kinetics has been described in the context of forced unfolding of ribozyme[52] and for the mechanical unraveling of proteins[79] and protein tandems.[80] This results in the formation of intermediate states en route to unfolding and leads to the emergence of multiple unfolding pathways. In addition, the extent of the force-induced decrease in barrier heights and motions of the transition states might vary for different pathways. This might change the pathway probabilities observed at different pulling speeds.

The results obtained for the *C2A* domain from human *Syt*1, and for the γ*C*-chain and double-*D* fragment from human Fb, carried out at v_f = 2.5 and 25 μm/s (SOP-GPU), and at v_f = 10^4–10^5 μm/s (all-atom MD), fully support the above arguments. We see that the global unfolding transitions in these protein systems follow

multiple kinetic pathways and involve formation of intermediate states (Figs. 3, 4, and 5), which is reflected in the force spectra. A moderate 10-fold increase in the pulling speed triggers the change in the mechanism of unfolding for the γ*C*-chain and for the double-*D* fragment leading to the simultaneous unfolding of various secondary structure elements. This is manifest in the force spectra, where, due to "kinetic disorder", the broader force signals become less resolved [Figs. 4(d) and 5(c)]. We have observed this effect both in SOP-GPU simulations and in SMD simulations at very fast force loads (Fig. 4). An even larger 10^4–10^5-fold increase in v_f results in substantial changes in pathway probabilities and in the emergence of a new pathway for the *C*2*A* domain [Fig. 3(d,e)]. For the γ*C*-chain, a 10^4–10^5-fold in v_f gives rise to an entirely different unfolding mechanism, not observed at the experimental values of v_f, where all the secondary structure elements unravel at the same time [Fig. 4(b)]. As a result, the fine structure of the force spectrum is reduced to a single peak [Fig. 4(e)].

In general, there are neither exactly solvable models nor analytically tractable approximations that can be used to describe the kinetics and thermodynamics of a protein for all values of an applied mechanical force. Under certain assumptions, one can use a simple scaling law for the dependence of the average unfolding force $\bar{f}$ on the pulling speed v_f, $\bar{f} \sim \ln[\kappa v_f]$. However, this relation assumes that the Bell model[81,82] applies in the entire range of pulling speeds used to unfold a given protein. In fact, the Bell model is valid only when the well-to-barrier distance is independent of force,[83,84] which is unlikely to be the case in the entire range of force loads from the experimental speeds of 1–10 μm/s to the values of 10^4–10^6 μm/s used in MD simulations. The force-induced motion of the transition state barrier is discussed, for example in Refs. 85, 86. Hence, in general, the dependence of $\bar{f}$ on v_f is rather complicated, and when the assumptions behind the Kramers theory[87,88] no longer hold, the scaling law, which can only be used when the unfolding pathway(s) remain(s) the same for varying v_f, is hard to derive.

All-atom MD simulations of the mechanical unfolding of proteins are crucial in pinpointing the structural underpinnings of the unfolding process.[89] Unfortunately, given the limited computational capacity available today, the pulling speeds (force-loading rates) used in MD simulations exceed their experimental counterparts by many orders of magnitude. This makes it difficult to directly relate the force signals, showing up in the experimental force-extension profiles, to the corresponding microscopic molecular changes observed in all-atom MD simulations. The SOP-GPU simulations, described in this work, allow for the theoretical inference of the kinetic pathways and structural features, underlying the micromechanics of unfolding, under the experimentally relevant conditions of force application.

CONCLUSION

Nanomanipulation of biomolecules by mechanical force is widely used to explore their physical properties and to control their energy landscape at the single-molecule level. There is a wealth of experimental data for a large number of biomolecular systems awaiting for the theoretical interpretation. The challenge is to extract the unfolding pathways and to resolve the molecular mechanism(s) from experimental measurements such as the force-extension curves and the force-indentation profiles. All-atomic modeling provides valuable information about the biomolecular transitions in the 1–100 μs timescale at the microscopic level. Yet, performing MD simulations of biological processes in implicit and explicit water on a CPU and on a GPU in the ms-s timescale, under experimentally relevant force loads, is virtually impossible even for a small system. This renders direct comparison of the results obtained from simulations and from dynamic force measurements difficult. Here, we developed the SOP-GPU software to characterize the mechanical denaturation of proteins and the physical properties of large-size protein assemblies using experimental force loads employed in AFM and laser-tweezer based dynamic force assays. We demonstrated that SOP-GPU simulations describe accurately the biomechanical unfolding reactions at the slow experimental pulling speeds (v_f = 1–10 μm/s), and that the results of SOP-GPU simulations agree well with the results of all-atom MD simulations at the faster force loads. The results obtained in this work show clearly the high predictive capacity of the simplified description of biomolecules empowered by advanced computations on graphics processors, which allow to span many decades of biological time.

The theoretical force-extension profiles, obtained from pulling simulations at the experimental pulling speeds v_f = 1–10 μm/s for several protein systems, including the *C*2*A* domain from human *Syt*1, and the γ*C*-chain and double-*D* fragment from human Fb, agree very well with the experimental force spectra. We showed that for these molecules the mechanical response depends rather sensitively on the pulling speed v_f and that the unfolding mechanisms and kinetic pathways change when v_f is increased. We would like to emphasize that this could be a general property shared by many biomolecules. Hence, for the accurate interpretation of the experimental data, dynamic force measurements *in silico* should be performed using the experimentally relevant pulling speeds. This can be achieved in reasonable wall-clock time using the SOP-GPU package. More detailed characterization of the mechanical properties of the *C*2*A* and *C*2*B* domains from human *Syt*1 will be presented in a separate work (manuscript in preparation). The results of the theoretical exploration of the molecular mechanisms underlying the force-induced elongation of the single-chain models of fibrin fibers $(Fb)_n$ will be given in a separate paper

(manuscript in preparation). Computational studies of the physical properties of viral capsids are under way (unpublished data).

In this work, we focused on the C_α-based coarse-grained model of a protein described by the two-body potential energy terms. However, the developed formalism can be used in conjunction with more sophisticated biomolecular force fields to explore, for example, protein-protein and protein-DNA interactions. The formalism can also be extended to include three-body (angle) potentials to describe side chains. The numerical integration kernel can be modified to follow the Langevin dynamics in the underdamped limit to study the thermodynamics of biomolecular transitions. Describing molecular mechanism(s) and multiple pathways, underlying biomolecular transitions, and resolving the entire distributions of the molecular characteristics requires gathering of statistically significant amount of data. This can be achieved employing the many-runs-per-GPU approach, which can also be used in parallel tempering algorithms, including variants of the Replica Exchange Method, to resolve the phase diagrams of biomolecules.

Due to the rapid evolution of the GPU hardware, the simulation time will decrease significantly on graphics processors with the MIMD architecture (Multiple Instruction Multiple Data), such as the new Fermi platform.[75] The CUDA program, tested in this work, will be able to run (with minor modifications) on this new hardware making use of the increased processing resources, and will permit the theoretical exploration of a range of interesting biological problems for which the experimental data are already available. The presented formalism can be applied to the studies of other systems, including molecular motors, proteasomes, nucleosomes, colloidosomes and liposomes. Beyond that, we note that the cost of one GeForce GTX 295 graphics card with 480 processors is $\sim$500, that is, $\sim$1 per processor, which makes GPUs a cost-efficient desktop-based alternative to the more expensive computer clusters.

ACKNOWLEDGMENTS

We thank Mr. L. Duan (Dima group) and Mrs. S. Agarwal (Barsegov group) for their help with running all-atom MD simulations.

REFERENCES

1. Stossel TP, Condeelis J, Cooley L, Hartwig JH, Noegel A, Schleicher M, Shapiro SS, Filamins as integrators of cell mechanics and signalling. Nat Rev Mol Cell Biol 2001;2:138–145.
2. Johnson CP, Tang HY, Carag C, Speicher DW, Discher DE. Forced unfolding of proteins within cells. Science 2007;317:663–666.
3. Paul R, Heil P, Spatz JP, Schwarz US. Propagation of mechanical stress through the actin cytoskeleton toward focal adhesions: model and experiment. Biophys J 2008;94:1470–1482.
4. Leckband D. Nanomechanics of adhesion proteins. Curr Opin Struct Biol 2004;14:523–530.
5. McEver RP. Selectins: lectins that initiate cell adhesion under flow. Curr Opin Cell Biol 2002;14:581–586.
6. Marshall BT, Long M, Piper JW, Yago T, McEver RP, Zhu C. Direct observation of catch bonds involving cell-adhesion molecules. Nature 2003;423:190–193.
7. Barsegov V, Thirumalai D. Dynamics of unbinding of cell adhesion molecules: Transition from catch to slip bonds. Proc Natl Acad Sci USA 2005;102:1835–1839.
8. Weisel JW. The mechanical properties of fibrin for basic scientists and clinicians. Biophys Chem 2004;112:267–276.
9. Weisel JW. Enigmas of blood clot elasticity. Science 2008;320:456–457.
10. Lord ST. Fibrinogen and fibrin: scaffold proteins in hemostasis. Curr Opin Hematol 2007;14:236–241.
11. Falvo MR, Washburn S, Superfine R, Finch M, Brooks Jr FP, Chi V, Taylor RM II. Manipulation of individual viruses: friction and mechanical properties. Biophys J 1997;72:1396–1403.
12. Uetrecht C, Verslius C, Watts NR, Roos WH, Wuite GJL, Wingfield PT, Steven AC, Heck AJ. High-resolution mass spectrometry of viral assemblies: molecular composition and stability of dimorphic hepatitis b virus capsids. Proc Natl Acad Sci USA 2008;105:9216–9220.
13. Kuznetsov YG, Daijogo S, Zhou J, Semler BL, McPherson A. Atomic force microscopy analysis of icosahedral virus RNA. J Mol Biol 2007;347:41–52.
14. Kol N, Shi Y, Tsvitov M, Barlam D, Shneck RZ, Kay MS, Rousso I. A stiffness switch in human immunodeficiency virus. Biophys J 2007;92:1777–1783.
15. Ivanovska IL, de Pablo PJ, Ibarra B, Sgalari G, MacKintosh FC, Carrascosa JL, Schmidt CF, Wuite GJL. Bacteriophage capsids: tough nanoshells with complex elastic properties. Proc Natl Acad Sci USA 2004;101:7600–7605.
16. Ivanovska I, Wuite G, Joensson B, Evilevitch A. Internal DNA pressure modifies stability of WT phage. Proc Natl Acad Sci USA 2007;104:9603–9608.
17. Steven AC, BHJ, Cheng N, Trus BL, Conway JF. Virus maturation: dynamics and mechanism of a stabilizing structural transition that leads to infectivity. Curr Opin Struct Biol 2005;15:227–236.
18. Carrion-Vazquez M, Li H, Lu H, Marszalek PE, Oberhauser AF, Fernandez JM. The mechanical stability of ubiquitin is linkage dependent. Nat Struct Biol 2003;10:738–743.
19. Schwaiger I, Sattler C, Hostetter DR, Rief M. The myosin coiled-coil is a trully elastic protein structure. Nature Mat 2002;1:232–235.
20. Brujic J, Hermans RI, Walther KA, Fernandez JM. Single-molecule force spectroscopy reveals signatures of glassy dynamics in the energy landscape of ubiquitin. Nature Phys 2006;2:282–286.
21. Brown AEX, Litvinov RI, Discher DE, Weisel JW. Forced unfolding of coiled-coils in fibrinogen by single-molecule AFM. Biophys J 2007;92:L39–L41.
22. Smith DE, Tans SJ, Smith SB, Grimes S, Anderson DL, Bustamante C. The bacteriophage ϕ 29 portal motor can package DNA against a large internal force. Nature 2001;413:748–752.
23. Roos WH, Ivanovska IL, Evilevitch A, Wuite GJL. Viral capsids: Mechanical characteristics, genome packaging and delivery mechanisms. Cell Mol Life Sci 2007;64:1484–1497.
24. Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M. CHARMM: a programm for macromolecular energy, minimization, and dynamics calculations. J Comp Chem 1983;4:187–217.
25. Phillips JC, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel RD, Kalé L, Schulten K. Scalable molecular dynamics with NAMD. J Comp Chem 2005;26:1781–1802.
26. Berendsen HJC, van der Spoel D, van Drunen R. GROMACS: a message-passing parallel molecular dynamics implementation. Comp Phys Commun 1995;91:43–56.
27. Isralewitz B, Gao M, Schulten K. Steered molecular dynamics and mechanical functions of proteins. Curr Opin Struct Biol 2001;11:224–230.

28. Stone JE, Phillips JC, Freddolino PL, Hardy DJ, Trabuco LG, Schulten K. Accelerating molecular modeling applications with graphical processors. J Comp Chem 2007;28:2618–2640.
29. Freddolino PL, Liu F, Gruebele M, Schulten K. Ten-microsecond MD simulation of a fast-folding WW domain. Biophys J 2008;94: L75–L77.
30. Zink M, Grubmueller H. Mechanical properties of the icosahedral shell of southern bean mosaic virus: A molecular dynamics study. Biophys J 2009;96:1767–1777.
31. AMD. ATI Stream Computing Technical Overview. AMD; Sunnyvale, CA, 2009.
32. NVIDIA. NVIDIA CUDA Programming Guide, 2.3.1 ed. NVIDIA; Santa Clara, CA, 2009.
33. NVIDIA. NVIDIA CUDA C Programming Best Practices Guide, 2.3 ed. NVIDIA; Santa Clara, CA, 2009.
34. Munshi A. Khronos OpenCL Working Group The OpenCL Specification, 1.0 ed. Beaverton, OR. Publisher-Khronos Group, 2009.
35. Anderson AG, Goddard WA, III, Schröder P. Quantum Monte Carlo on graphical processing units. Comp Phys Commun 2007;177:298–306.
36. Yasuda K. Two-electron integral evaluation on the graphics processor unit. J Comput Chem 2007;29:334–342.
37. Yasuda K. Accelerating density functional calculations with graphics processing Unit. J Chem Theor Comput 2008;4:1230–1236.
38. Ufimtsev IS, Martinez TJ. Quantum chemistry on graphics processing units. 1. Strategies for two-electron integral evaluation. J Chem Theor Comput 2008;4:222–231.
39. Vogt L, Olivares-Amaya R, Kermes S, Shao Y, Amador-Bedolla C, AspuruGuzik A. Accelerating resolution-of-the-identity second-order Moller-Plesset quantum chemistry calculations with graphical processing units. J Phys Chem A 2008;112:2049–2057.
40. Rodrigues CI, Hardy DJ, Stone JE, Schulten K, Hwu W-MW. GPU acceleration of cutoff pair potentials for molecular modeling applications. CF ′08: Proceedings of the 5th conference on computing frontiers, New York, NY, USA. ACM; 2008;273–282.
41. Phillips JC, Stone JE, Schulten K. Adapting a message-driven parallel application to GPU-accelerated clusters. SC ′08: Proceedings of the 2008 ACM/IEEE conference on supercomputing, Piscataway, NJ, USA: IEEE Press; 2008. pp. 1–9.
42. Friedrichs MS, Eastman P, Vaidyanathan V, Houston M, Legrand S, Beberg AL, Ensign DL, Bruins CM, Pande VS. Accelerating molecular dynamic simulation on Graphics Processing Units. J Comput Chem 2009;30:864–872.
43. Harvey MJ, Giupponi G, Fabritilis GD. ACEMD: accelerating biomolecular dynamics in the microsecond time scale. J Chem Theory Comput 2009;5:1632–1639.
44. Harvey MJ, Fabritilis GD. An implementation of the smooth Particle Mesh Ewald method on GPU hardware. J Chem Theory Comput 2009;5:2371–2377.
45. Davis JE, Ozsoy A, Patel S, Taufer M. Towards large-scale molecular dynamics simulations on graphics processors. BICoB ′09: Proceedings of the 1st international conference on bioinformatics and computational biology, Berlin, Heidelberg. Springer-Verlag: New York, NY, 2009;176–186.
46. van Meel JA, Arnold A, Frenkel D, Zwart SFP, Belleman R. Harvesting graphics power for MD simulations. Mol Simul 2008;34:259–266.
47. Anderson JA, Lorentz CD, Travesset A. General purpose molecular dynamics simulations fully implemented on graphics processing units. J Comp Phys 2008;227:5342–5359.
48. Tozzini V. Coarse-grained models for proteins. Curr Opin Struct Biol 2005;15:144–150.
49. Clementi C, Nymeyer H, Onuchic JN. Toplogical and energetic factors: what determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding?. J Mol Biol 2000;298:937–953.
50. Veitshans T, Klimov D, Thirumalai D. Protein folding kinetics: timescales, pathways and energy landscapes in terms of sequence-dependent properties. Folding and Design 1997;2:1–22.
51. Klimov DK, Thirumalai D. Native topology determines force-induced unfolding pathways in globular proteins. Proc Natl Acad Sci USA 2000;97:7254–7259.
52. Hyeon C, Dima RI, Thirumalai D. Pathways and kinetic barriers in mechanical unfolding and refolding of RNA and proteins. Structure 2006;14:1633–1645.
53. Mickler M, Dima RI, Dietz H, Hyeon C, Thirumalai D, Rief M. Revealing the bifurcation in the unfolding pathways of GFP using single molecule experiments and simulations. Proc Natl Acad Sci USA 2007;104:20268–20273.
54. Dima RI, Joshi H. Probing the origin of tubulin rigidity with molecular simulations. Proc Natl Acad Sci USA 2008.
55. Hyeon C, Onuchic JN. Internal strain regulates the nucleotide binding site of the kinesin leading head. Proc Natl Acad Sci USA 2007;104:2175–2180.
56. Lin JC, Thirumalai D. Relative stability of helices determines the folding landscape of adenine riboswitch aptamers. J Am Chem Soc 2008;130:14080–14084.
57. Hyeon C, Lorimer GH, Thirumalai D. Dynamics of allosteric transitions in GroEL. Proc Natl Acad Sci USA 2006;103:18939–18944.
58. Chen J, Dima RI, Thirumalai D. Allosteric communication in dihydrofolate reductase: signaling network and pathways for closed to occluded transition and back. J Mol Biol 2007;374:250–266.
59. Hyeon C, Jennings PA, Adams JA, Onuchic JN. Ligand-induced global transitions in the catalytic domain of protein kinase A. Proc Natl Acad Sci USA 2009;106:3023–3028.
60. Tehver R, Thirumalai D. Rigor to post-rigor transition in myosin V: Link between the dynamics and the supporting architecture. Structure 2010;18:471–481.
61. Kirk DB, Wen-Mei WH. Programming Massively Parallel Processors: a hands-on approach. Morgan Kaufmann: Burlington, MA; 2010.
62. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Sindyalov IN, Bourne PE. The protein databank. Nucleic Acids Res 2000;28:235–242.
63. Levesque D, Verlet L, Kürkijarvi J. Computer "experiments" on classical fluids. IV. Transport properties and time-correlation functions of the Lennard-Jones liquid near its triple point. Phys Rev A 1973;7:1690–1700.
64. Box GEP, Miller ME. A note on the generation of normal random deviates. Ann Math Stat 1958;29:610–611.
65. Nguyen H, editor. GPU Gems 3. Addison-Wesley; Boston, MA; 2008.
66. Carrillo-Tripp M, Shepherd CM, Borelli IA, Venkataraman S, Lander G, Natarajan P, Johnson JE, Brooks ICL, Reddy VS. VIPERdb2: an enhanced and web API enabled relational database for structural virology. Nucl Acid Res 2009;37:D436–D442.
67. Ferguson N, Johnson CM, Macias M, Oschkinat H, Fersht AR. Ultrafast folding of WW domains without structured aromatic clusters in the denatured state. Proc Natl Acad Sci USA 2001;98:13002–13007.
68. Karanicolas J, Brooks CL. Structural basis for biphasic kinetics in the folding of the WW domain from a formin-binding proteins: lessons for protein design? Proc Natl Acad Sci USA 2003;100:3954–3959.
69. Ermak DL, McCammon JA. Brownian dynamics with hydrodynamic interactions. J Chem Phys 1978;69:1352–1360.
70. Rief M, Gautel M, Oesterhelt F, Fernandez JM, Gaub HE. Reversible unfolding of individual titin immunoglobulin domains by AFM. Science 1997;276:1109–1112.
71. Carrion-Vazquez M, Oberhauser AF, Fisher TE, Marszalek PE, Li H, Fernandez JM. Mechanical design of proteins studied by single-molecule force spectroscopy and protein engineering. Prog Biophys Mol Biol 2000;74:63–91.
72. Martens S, Kozlov MM, McMahon HT. How synaptotagmin promotes membrane fusion. Science 2007;316:1205–1208.
73. Fuson KL, Ma L, Sutton RB, Oberhauser AF. The C2 domains of human synaptotagmin 1 have distinct mechanical properties. Biophys J 2009;96:1083–1090.

74. Yang J, Wang Y, Chen Y. GPU accelerated molecular dynamics simulations of thermal conductivities. J Comp Phys 2007;221:799–804.
75. NVIDIA. NVIDIA'S Next generation CUDA Compute Architecture: Fermi, 1.1 ed. NVIDIA; Santa Clara, CA, 2009.
76. Marszalek P, Lu H, Li H, Carrion-Vazquez M, Oberhauser A, Schulten K, Fernandez J. Mechanical unfolding intermediates in titin modules. Nature 1999;402:100–103.
77. Michel JP, Ivanovska IL, Gibbons MM, Klug WS, Knobler CM, Wuite GJL, Schmidt CF. Nanoindentation studies of full and empty viral capsids and the effects of capsid protein mutations on elasticity and strength. Proc Natl Acad Sci USA 2006;103:6184–6189.
78. Barsegov V, Morrison G, Thirumalai D. Role of internal chain dynamics on the rupture kinetics of adhesive contacts. Phys Rev Lett 2008;100:248102–248105.
79. Barsegov V, Klimov DK, Thirumalai D. Mapping the energy landscape of biomolecules using single molecule force correlation spectroscopy: theory and applications. Biophys J 2006;90:3827–3841.
80. Bura E, Klimov DK, Barsegov V. Analyzing forced unfolding of protein tandems by ordered variates, 1: independent unfolding times. Biophys J 2007;93:1100–1115.
81. Bell GI. Models for the specific adhesion of cells to cells. Science. 1978;200:618–627.
82. Evans E, Ritchie K. Dynamic strength of molecular adhesion bonds. Biophys J 1997;72:1541–1555.
83. Hummer G, Szabo A. Kinetics from nonequilibrium single-molecule pulling experiments. Biophys J 2003;85:5–15.
84. Dudko OK, Hummer G, Szabo A. Theory, analysis, and interpretation of single-molecule force spectroscopy experiments. Proc Natl Acad Sci USA 2008;105:15755–15760.
85. Hyeon C, Thirumalai D. Forced-unfolding and force-quench refolding of RNA hairpins. Biophys J 2006;90:3410–3427.
86. Hyeon C, Thirumalai D. Measuring the energy landscape roughness and the transition state location of biomolecules using single molecule mechanical unfolding experiments. J Phys Cond Matt 2007;19:113101–113128.
87. Zwanzig R. Nonequilibrium statistical mechanics. Oxford, U.K.: Oxford University Press: 2001.
88. Hanggi P, Talkner P, Borkovec M. Reaction-rate theory: fifty years after Kramers. Rev Mod Phys 1990;62:251–341.
89. Lu H, Schulten K. Steered molecular dynamics simulations of force-induced protein domain unfolding. Proteins 1999;35:453–463.