МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ 2011 год, том 23, номер 10, стр. 133-156

УДК 519.672-519.674

МОДЕЛИРОВАНИЕ МИКРОМЕХАНИКИ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ С ИСПОЛЬЗОВАНИЕМ ДИНАМИКИ ЛАНЖЕВЕНА

© *2011 г.* ***А.А. Жмуров, В.А. Барсегов, С.В. Трифонов, Я.А. Холодов, А.С. Холодов***

Московский физико-технический институт
kholodov@crec.mipt.ru

Работа проведена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы и поддержана грантом РФФИ № 09-07-12132.

Поскольку основные биологические процессы, такие как сворачивание белка и механическая денатурация белка, происходят на достаточно длинном временном интервале (от микросекунды до секунды), теоретическое моделирование поведения биомолекул в реальных физиологических условиях представляет собой сложную задачу даже для распределенных вычислительных систем. За последние несколько лет графические процессоры (ГП) эволюционировали в альтернативную традиционным центральным процессорам (ЦП) вычислительную платформу. Благодаря тому, что современные ГП могут обеспечить несравнимо высокую (относительно ЦП) вычислительную мощность, они сейчас используются в широком спектре научных приложений. Используя упрощённую модель самоорганизующегося полимера (Self Organized Polymer, SOP), мы разработали и протестировали программную реализацию моделирования механического поведения белков с использованием динамики Ланжевена на базе ГП (программа SOP-GPU), в которой все вычислительные шаги алгоритма были перенесены на ГП. Параллельное вычисление сил взаимодействия частиц было реализовано двумя различными способами: распараллеливанием по частицам и по взаимодействующим парам частиц. Эффективное использование текстурного кэша и аппаратного ускоренния математических функций позволило добиться 90-кратного ускорения по сравнению с оптимизированной версией той же программы для ЦП. Мы оценили вычислительную производительность программной реализации SOP-GPU, то есть время выполнения программы и загрузку памяти, выполняя расчёты для небольших белков, длинных белковых волокон и больших белковых комплексов. Разработанную реализацию программы SOP-GPU можно использовать для теоретического исследования механических свойств больших белковых систем и для получения профилей растяжения и сжатия в экспериментальных условиях приложения силы. Программа также может быть использована для прямого сопоставления результатов экспериментов на единичных молекулах *in vitro* и *in silico*.

Ключевые слова: графические процессоры, моделирование поведения белковых систем, модель самоорганизующегося полимера, динамика Ланжевена, программа SOP-GPU.

LANGEVIN DYNAMICS SIMULATIONS OF MICROMECHANICS ON GRAPHICS PROCESSORS

A.A. Zhmurov, V.A. Barsegov, S.V. Trifonov, Y.A. Kholodov, A.S. Kholodov

Moscow Institute of Physics and Technology

Due to the very long timescales involved (*ms*−*s*), theoretical modeling of fundamental biological processes including folding, misfolding, and mechanical unraveling of biomolecules, under physiologically relevant conditions, is challenging even for distributed computing systems. Graphics Processing Units (GPUs) are emerging as an alternative programming platform to the more traditional CPUs as they provide high raw computational power that can be utilized in a wide range of scientific applications. Using a coarse-grained Self Organized Polymer (SOP) model, we have developed and tested the GPU-based implementation of Langevin simulations for proteins (SOP-GPU program). Simultaneous calculation of forces for all particles is implemented using either the particle based or the interacting pair based parallelization, which leads to a ~90-fold acceleration compared to an optimized CPU version of the program. We assess the computational performance of an end-to-end application of the SOP-GPU program, where all steps of the algorithm are running on the GPU, by profiling the associated simulation time and memory usage for a number of small proteins, long protein fibers, and large-size protein assemblies. The SOP-GPU package can now be used in the theoretical exploration of the mechanical properties of large-size protein systems to generate the force-extension and force-indentation profiles under the experimental conditions of force application, and to relate the results of single-molecule experiments *in vitro* and *in silico*.

Key words: graphics processing units, large-size protein systems simulations, self organized polymer model, Langevin dynamics, SOP-GPU package.

1. Введение

Белковые волокна, например, фибронектин, волокна фибрина, микротрубочки и актиновые филаменты, выполняют важные механические функции при формировании цитоскелета и поддержании работоспособности клетки [1–3] в процессе слияния клеток и формировании внеклеточной матрицы [4–7], а также при свертывании крови [8–10]. Физические свойства капсул растительных и животных вирусов [11–13], ретровирусов [14] и бактериофагов [15,16], а также переходы между их стабильным и нестабильным состояниями определяют жизненный цикл многих вирусов, в том числе созревание вируса и заражение клеток [17]. Изучение происхождения уникальных вязкоэластичных свойств белковых волокон и механизмов перехода от эластичного к пластичному состоянию в капсулах, а также возможность контролировать их динамическое поведение в ответ на механическое воздействие являются важными сферами исследования в биофизике. Современные технологии, предназначенные для изучения одиночных молекул, такие как атомно-силовая микроскопия и оптические пинцеты, широко используются для экспериментального изучения механических свойств белковых волокон [18–21] и капсул вирусов [15,16,22,23]. Однако ввиду сложности строения этих систем (~10^3–10^5 частиц) и их больших размеров (~50–200 *нм*), результаты подобных экспериментов почти невозможно интерпретировать без предварительных знаний о ландшафте свободной энергии [9].

Стандартные вычислительные пакеты молекулярной динамики (МД) в полноатомном разрешении, такие как CHARMM [24], NAMD [25] и Gromacs [26], широко используются для изучения поведения биомолекул на субмолекулярном уровне. Так как полноатомное моделирование в настоящее время ограничено размером молекулы в 10–50 *нм* и длительностью процесса моделирования в 0.1–10 *мс* [27-29], данный подход хорошо подходит только для моделирования равновесных процессов, а достижение биологически важного временного интервала от микросекунды до секунды практически невозможно даже для малых систем. Что ещё более важно, для подробного изучения ландшафта свободной энергии, лежащего в основе изучаемого биологического процесса, требуется статистически значимое количество траекторий. Одно из возможных решений такой задачи – проведение МД моделирования на компьютерных кластерах – требует огромных вычислительных ресурсов и длительного времени выполнения программы. К примеру, для расчёта 20 коротких (1 *нс*) траекторий для южного вируса мозаики бобовых (southern bean mosaic virus), состоящего из более чем 4.5×10^6 атомов, потребовалось 800 000 процессоро-часов работы кластера SGI Altix 4700 [30]. Это ограничивает возможность применения вычислительного эксперимента для изучения широкого спектра биологических проблем, таких как деформация волокон белков, формирование биомолекулярных комплексов и агрегатов, механическое повреждение капсул вирусов, для которых экспериментальные данные уже получены, а прямое сопоставление результатов экспериментов и вычислительных расчётов невозможно.

Хотя графические процессоры изначально были спроектированы для ускорения работы с трёхмерной графикой, современные ГП способны выполнять многие вычислительные задачи, в том числе и те, которые не связаны с обработкой изображения. Недавние технологические достижения на аппаратном уровне, поддержка стандарта IEEE для вещественной арифметики позволяет использовать огромные вычислительные возможности ГП в научных приложениях. В отличие от процессоров с привычной архитектурой, большинство логических элементов ГП отведено на выполнение вычислений, а не на кэш-память и управление логикой. Массивная многопоточность, минимальный контекст потоков и высокая пропускная способность памяти делают ГП эффективным массивно-параллельным вычислительным устройством (рис.1). Программные платформы для современных ГП включают ATI Stream Computing [31], NVIDIA Compute Unified Device Architecture (CUDA) [32, 33] и Open Computing Language (OpenCL) [34]. CUDA, программная среда для параллельных вычислений, является высокоуровневой программной платформой, расширяющей стандартные языки C и C++. Это позволяет разработчику реализовывать процедуры (ядра), которые могут выполняться одновременно во множестве независимых потоков на ГП.

Из-за фундаментальных различий архитектуры ГП и ЦП методы молекулярного моделирования, разработанные для выполнения на ЦП, не могут быть просто перенесены или адаптированы для работы на ГП. Тем не менее, в молекулярной динамике парные взаимодействия обычно описываются одной и той же эмпирической функцией потенциальной энергии для всех пар взаимодействующих частиц (силовое поле), а динамика системы определяется из численного решения одного и того же уравнения движения для всех частиц. Таким образом, существует прямое соответствие между SIMD (Single Instruction, Multiple Data) архитектурой ГП на аппаратном уровне и вычислительными процедурами молекулярной динамики на программном уровне. Можно выполнить

одну и ту же процедуру (вычисление потенциальной энергии или сил, генерация случайных чисел и численное интегрирование уравнений движения) одновременно для различных наборов данных (для всех частиц) за счёт использования множества арифметико-логических устройств, работающих параллельно, повторяя вычисления на протяжении множества шагов по времени. Поэтому молекулярные расчеты – естественный кандидат для реализации на ГП, но для эффективного выполнения алгоритма на ГП он должен быть преобразован для запуска множества независимых потоков, выполняющих одинаковый программный код на различных наборах данных одновременно. На данный момент существуют предварительные версии стандартных пакетов для моделирования МД белков, реализованные на ГП, такие как NAMD [28, 35, 36], Gromacs [47] и др. [38-40]. Однако до сих пор нет реализации процесса моделирования поведения биомолекул с использованием динамики Ланжевена. В этой работе мы представляем и тестируем первую такую реализацию. Так как молекулярные преобразования в белках, подверженных внешнему механическому воздействию, определяются главным образом топологией и общей структурой системы, для описания белков мы используем упрощённую [41-43] модель самоорганизующегося полимера (Self Organized Polymer, SOP) [44,45].

Реализация алгоритмов динамики Ланжевена на ГП требует понимания архитектуры вычислительного устройства. В следующей главе представлена методология реализации процесса молекулярного моделирования по принципам динамики Ланжевена на ГП. Она включает описание двух основных подходов к распараллеливанию вычисления сил: по частицам и по взаимодействующим парам, а также процедуры генерации псевдослучайных чисел, создание списков Верле и численное решение уравнений движения Ланжевена. Аналогичные численные методы могут быть использованы и в программной реализации молекулярного моделирования в полноатомном разрешении. Для того чтобы сделать описание алгоритмов более доступным, мы старались упростить описание формализма задачи, сфокусировав большее внимание на наиболее важных вычислительных аспектах. В главе 3 приводится сравнительный анализ результатов моделирования процесса механической денатурации белков на ЦП и ГП, а также оценка точности численного интегрирования, произведенные при помощи тестовой системы – домена *WW*. Приводятся результаты моделирования на ГП в терминах времени, затраченного на моделирование, требований к памяти и ускорения вычислений (на ГП по сравнению с ЦП) для ряда белков, включая небольшие белки (*WW*-домен, *Ig*27-домен из титина человека, *C*2*A*-домен из синаптотагмина человека (*Syt*1), γ*C*-цепь и фрагмент *D*-димер из фибриногена человека (*Fb*)), одноцепочные модели белковых волокон фибрина (Fb мономер и димер) и большие белковые образования (капсула вируса *HK*97). Наиболее значимые из полученных результатов перечислены в разд. 4.

2. Моделирование по принципам динамики Ланжевена на ГП

В данном разделе описываются методы параллельных вычислений потенциалов и сил парного взаимодействия между частицами, основанные на подходах к распараллеливанию по отдельным частицам или по взаимодействующим парам частиц. Также представлены численные методы для создания списков Верле, генерации псевдослучайных чисел и интегрирования уравнений движения Ланжевена. Разработанный нами алгоритм рассчитан на то, чтобы одновременно выполнять на ГП максимально возможное число вычислительных процедур. Это позволяет добиться высокого уровня эффективно-

сти работы ГП и минимизировать взаимодействие между ГП и ЦП. Структурное разбиение алгоритма и порядок выполнения вычислений ГП и ЦП представлены на рис.1, где также показаны потоки обмена данными между оперативной памятью ЦП и глобальной памятью ГП.

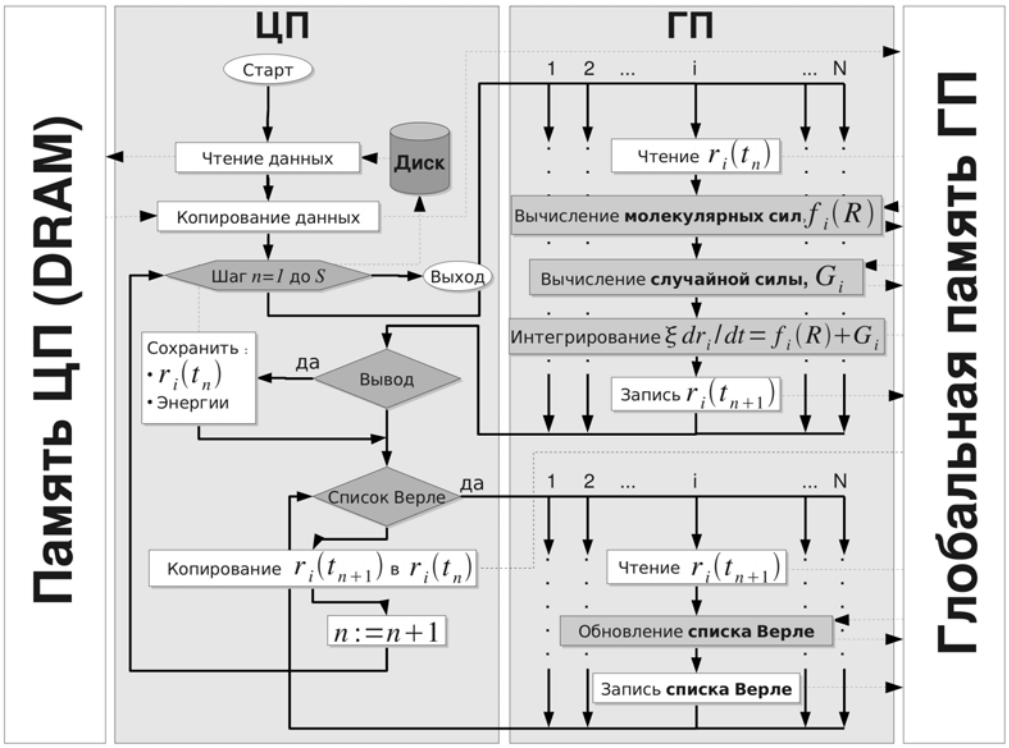

Рис.1. Схема работы алгоритма и вычислительных процедур для молекулярной симуляции с использованием динамики Ланжевена на ГП. Чёрные стрелки показывают переходы между вычислительными процедурами. Передача данных между ЦП, ГП, а также сохранение данных на диске, показаны пунктирными стрелками. Вычислительные процедуры на ГП показаны только для одного потока (с идентификатором *i*), также детально показано разделение вычислительной нагрузки между ЦП и ГП. Выполнение программы начинается на ЦП, который используется для первичной обработки данных, контроля вычислительных процедур и сохранения результатов. Все вычисления, связанные с расчетом сил, интегрированием уравнений движения и обновлением списка Верле выполняются на ГП во многих независимых потоках одновременно.

При численном моделировании динамики Ланжевена молекулярные силы обычно описываются парными потенциалами: гармоническим потенциалом, потенциалом FENE (finitely extensible nonlinear elastic potential) [46], потенциалом Леннард-Джонса и т.д., которые отличаются только математической формулой. Следовательно, один и тот же обобщённый алгоритм может использоваться для расчёта всех этих потенциалов. В последующих псевдокодах каждая из N аминокислот белка имеет уникальный индекс

$i \in [0, N-1]$, массив r – массив координат частиц, где $r[i]$ – координаты i-й частицы. Мы также используем следующие обозначения: чтение из глобальной памяти ГП обозначено как '⇐'; сохранение данных в глобальной памяти ГП – '⇒'; '<=' указывает на чтение данных из кэша; '←' означает обращение к локальной, разделяемой или постоянной памяти, или присваивание значения переменной.

***Алгоритм 1*:** Расчёт сил взаимодействия между частицами (реализация для ЦП)

1. $P \leftarrow$ общее число пар в системе для данного потенциала
2. **for** $p = 0$ to $P - 1$ **do**
3. $i \leftarrow pairs[p].i$ {индекс первой частицы пары}
4. $j \leftarrow pairs[p].j$ {индекс второй частицы пары}
5. $par \leftarrow pairs[p].parameters$ {параметры для i–j пары}
6. $r \leftarrow r[i] - r[j]$
7. $df \leftarrow$ force (r, par)
8. $f[i] \leftarrow f[i] + df$
9. $f[j] \leftarrow f[j] - df$
10. **end for**

Информация обо всех парах аминокислот сохраняется в массиве *pairs*, элементы которого описываются индексами i, j и переменной *par* для постоянных параметров, описывающих потенциальную энергию. В основном цикле на ЦП (строка 2), индекс p проходит по всем парам взаимодействующих частиц $p = 0, 1, \ldots, P - 1$. Для каждой пары собирается информация (строки 3–5) о координатах ($r[i]$ и $r[j]$) и постоянных параметрах (*par*). Далее с помощью функции force (…) вычисляется сила взаимодействия *df* для данной пары p (строка 7). Это значение прибавляется к результирующей силе, действующей на i-ю частицу ($f[i]$, строка 8), и вычитается из значения результирующей силы, действующей на j-ю частицу ($f[j]$, строка 9). На ЦП значения сил вычисляются только один раз (строки 7–9). Так как эти вычисления происходят последовательно, нет необходимости отслеживать их перекрывание во времени. Для сравнения, на ГП потенциал для разных пар аминокислот вычисляется в разных потоках, и простое сложение (строки 8 и 9) может вызвать конфликт доступа к памяти, когда несколько потоков обращаются к одному адресу глобальной памяти ГП одновременно.

Есть два основных подхода, позволяющих избежать ситуации конфликта доступа к памяти на ГП. Первый подход предполагает, что все значения сил для одной частицы вычисляются в одном потоке. Вычисление всех сил, действующих на все частицы, в этом случае требует запуска N потоков. Мы называем этот метод «распараллеливанием по частицам». Использование такого подхода приводит к тому, что одна и та же сила, действующая на i-ю и j-ю частицы, вычисляется дважды в i-м и j-м потоках. Другой подход, который мы называем «распараллеливанием по взаимодействующим парам», выполняет расчёт сил взаимодействия для всех пар частиц одновременно в P независимых потоках. $2P$ значений сил сохраняются по разным адресам глобальной памяти ГП. Далее мы более детально остановимся на обеих оптимизационных стратегиях, позволяющих максимально использовать вычислительные возможности ГП.

2.1. Распараллеливание по частицам. При использовании этого подхода вычисления происходят в N независимых потоках, выполняющихся на ГП одновременно. Ка-

ждый из потоков вычисляет значения потенциальной энергии и силы для всех взаимодействующих пар, включающих конкретную частицу. Таким образом, в N потоках независимо друг от друга, вычисляются N сил, действующих на N частиц. Несмотря на то что в этом случае сила, действующая между i-й и j-й частицами, вычисляется дважды (в i-м и j-м потоках), количество обращений к глобальной памяти при использовании такого подхода минимально, так как силы суммируются в локальной памяти сразу после расчёта. Таким образом, время, потраченное на вычисление одной и той же силы дважды, компенсируется за счёт экономии времени ожидания при чтении и записи значений силы в глобальную память ГП.

***Алгоритм 2*:** Расчёт сил взаимодействия между частицами (реализация для ГП с использованием подхода распараллеливания по частицам)

1. $f_i \leftarrow 0$ {результирующая сила}
2. $i \leftarrow$ индекс потока вычислений на ГП {равен индексу частицы}
3. $r_i <= r[i]$ {координаты i-й частицы}
4. $P_i \Leftarrow P_p[i]$ {число пар с участием i-й частицы}
5. **for** $p = 0$ to $P_i - 1$ **do**
6. $j \Leftarrow PairsMap[i][p].j$ {вторая частица пары}
7. $r_j <= r[j]$ {координаты j-й частицы}
8. $par \Leftarrow PairsMap[i][j].parameters$ {параметры пары i-й и j-й частиц}
9. $r \leftarrow r_i - r_j$
10. $df \leftarrow$ force (r, par)
11. $f_i \leftarrow f_i + df$
12. **end for**
13. Output: f_i

Массив P_p, в котором хранится количество пар для каждой из частиц, и матрица *PairsMap* всех пар частиц заранее вычисляются на ЦП и передаются в общую память ГП. P_p представляет собой N-мерный целочисленный вектор. Каждый элемент этого вектора соответствует одной частице, и i-е значение показывает число других частиц, с которыми взаимодействует i-я частица. *PairsMap* – это двумерная матрица размером $N \times M$, где M – максимальное число из массива P_p. i-я строка матрицы *PairsMap* соответствует i-й частице и содержит индексы всех частиц, взаимодействующих с ней, а также постоянные параметры функции потенциальной энергии. Данные в массиве *PairsMap* могут легко быть представлены таким образом, чтобы потоки обращались к смежным областям памяти, выполняя условие коалесинга. Обращения к координатам второй частицы пары происходят в произвольном порядке, что может привести к существенным потерям производительности. Чтобы сократить эти потери, можно использовать доступ к глобальной памяти графической карты через текстурные ссылки, что существенно сокращает количество обращений к глобальной памяти за счёт эффективного использования текстурного кэша.

Значения сил вычисляются параллельно в N потоков описанным далее образом. Вначале i-й поток считывает из глобальной памяти координаты i-й частицы, r_i (строка 3), и число пар с участием этой частицы $P_p[i]$ (строка 4). Проходя в цикле по всем парам, содержащим i-ю частицу (строки 5–12), поток определяет по индексу j координаты r_j второй частицы пары и постоянные параметры функции потенциальной энергии *par*

(строки 6–8). Эти данные далее используются для вычисления значения силы *df* (строка 10), которое добавляется к результирующей силе *f* (строка 11), действующей на *i*-ю частицу. Параметры *par* зависят от типа вычисляемого потенциала. Например, для ковалентной связи, описываемой гармоническим потенциалом $V_H = K_{ij}^{sp}(r_{ij} - r_{ij}^0)^2/2$, *par* содержит равновесное расстояние r_{ij}^0 и постоянную пружины K_{ij}^{sp}.

Списки Верле: При моделировании молекулярных систем информация о ковалентных связях и нативных взаимодействиях (массив P_p и матрица *PairsMap*), обычно получаемая из кристаллической структуры белка, не меняется. Однако, при вычислении дальнодействующих потенциалов, таких как электростатические взаимодействия и силы Ван дер Вальса, информацию о взаимодействиях между частицами нужно периодически обновлять. Это делается для того, чтобы сократить количество вычислений, так как сложность вычислений потенциала взаимодействия порядка $O(N^2)$ является наиболее вычислительно затратной частью алгоритма. Обычный подход для оптимизации расчета дальнодействующих потенциалов основан на том, что сила взаимодействия стремится к нулю с увеличением расстояния между частицами. Это позволяет ограничить список пар взаимодействующих частиц только теми, которые находятся в пределах заданного расстояния (списки Верле) [47]. Распараллеливание по частицам предполагает, что массив P_p и матрица *PairsMap* периодически перестраиваются, что позволяет ускорить вычисление сил взаимодействия, пренебрегая силами, действующими между сильно удалёнными частицами. На ГП алгоритм построения списков Верле можно получить путём преобразования алгоритма расчёта межчастичных сил (Алгоритм 2), как это показано в Алгоритме 3:

***Алгоритм 3*:** Построение списков Верле при использовании метода распараллеливания по частицам.

1. $i \leftarrow$ индекс потока вычислений на ГП {равен индексу частицы}
2. $p_i \leftarrow 0$ {счётчик пар частиц в списке Верле}
3. $r_i <= r[i]$ {координаты *i*-й частицы}
4. $P_{p,i} \Leftarrow P_{pp}[p_1]$ {число пар с участием *i*-й частицы}
5. **for** $p_p = 0$ to $P_{p,i} - 1$ **do**
6. $j \Leftarrow PossiblePairsMap[i][p_p]$ {вторая частица пары}
7. $r_j <= r[j]$ {координаты *j*-й частицы}
8. $r \leftarrow | r[i] - r[j] |$
9. **if** $r < cutoff$ **then**
10. PossiblePairsMap[i][p_p] ⇒PairsMap[i][p_i]
11. $p_i \leftarrow p_i + 1$
12. **end if**
13. **end for**
14. $p_i \Rightarrow P_p[i]$

Проход по циклу по p_p позволяет из всех возможных взаимодействующих пар выбрать пары, находящиеся в радиусе взаимодействия. Расстояния между частицами рассчитываются в строке 8. Количество частиц p_i, находящихся в радиусе взаимодействия, определяется в строке 11. Найденные новые пары добавляются в *PairsMap*, то есть копируются из матрицы *PossiblePairsMap* (все возможные пары) в матрицу *PairsMap* (под-

ходящие пары) на позицию (i, p_i). Когда цикл пройден, значение p_i, равное числу взаимодействующих пар списка Верле, сохраняется в массиве $P_p[i]$.

2.2. Распараллеливание по парам взаимодействующих частиц. Для того чтобы избежать двойного расчёта сил для парных потенциалов на ГП, можно использовать другой алгоритм, в котором каждый поток рассчитывает одно значение потенциала для пары частиц. При этом силы, разнонаправленно действующие на взаимодействующие частицы, вычисляются лишь один раз, а полученные значения сохраняются в различных ячейках глобальной памяти ГП. Затем все силы, действующие на отдельную частицу, суммируются для вычисления результирующей силы. Использование этого подхода требует дополнительных обращений к памяти и разработки отдельной процедуры, выполняющей итоговое суммирование. Тем не менее, он позволяет ускорить выполнение моделирования, если количество частиц N того же порядка, что и число АЛУ и/или если вычисление парных потенциалов ресурсозатратно. В следующем фрагменте псевдокода алгоритма для расчёта сил P (число потоков) равно количеству взаимодействующих пар (только для одного слагаемого потенциальной энергии), и каждый поток вычисляет силы для отдельной пары частиц:

***Алгоритм 4*:** Расчёт сил взаимодействия между частицами (реализация для ГП с использованием подхода распараллеливания по взаимодействующим парам частиц)

1. $p \leftarrow$ индекс потока вычислений на ГП {равен индексу пары}
2. $pair \Leftarrow pairs[p]$
3. $pair \Leftarrow PairsParameters[p]$ {параметры одной пары частиц p}
4. $i \leftarrow pair.i$ {i-я частица пары}
5. $j \leftarrow pair.j$ {j-я частица пары}
6. $shift_i \leftarrow pair.shift_i$ {позиция в массиве сил для i-й частицы}
7. $shift_j \leftarrow pair.shift_j$ {позиция в массиве сил для j-й частицы}
8. $r_i <= r[i]$ {координаты i-й частицы}
9. $r_j <= r[j]$ {координаты j-й частицы}
10. $r \leftarrow r_i - r_j$
11. $df \leftarrow$ force (r, par)
12. $df \Rightarrow F[i][shift_i]$ {сохраняем значение силы для i-й частицы}
13. $-df \Rightarrow F[j][shift_j]$ {сохраняем значение силы для j-й частицы}

Каждому потоку соответствует одна пара с тем же индексом p, который идентифицирует поток. Потоком считывается информация о потенциале из вектора *pairs* и о значениях постоянных параметров *par* из вектора *PairsParameters*, определяются взаимодействующие частицы i и j, их координаты (r_i и r_j), а также адреса глобальной памяти, по которым необходимо сохранять значения сил ($shift_i$ и $shift_j$). В массиве F значения сил сохраняются на позициях, соответствующих индексу частицы i и j (i-я или j-я строка) и параметрам $shift_i$ и $shift_j$ (столбцы). В массиве *pairs* итоговые позиции i и $shift_i$ для i-й частицы должны быть уникальны для каждой из частиц с тем, чтобы вычисленные значения сил сохранялись в глобальной памяти ГП по различным адресам. Это позволяет избежать конфликтов памяти, но требует разработки дополнительного ядра, выполняющего суммирование всех сил, действующих на определённую частицу, которые хранятся в массиве F (Алгоритм 4):

***Алгоритм 5*: Ядро сборки для вычисления результирующей силы**

1. $i \leftarrow$ индекс потока вычислений на ГП {равен индексу частицы}
2. $f_i \leftarrow 0$ {результирующая сила (определяется одним слагаемым потенциала)}
3. $P_i \Leftarrow P_p[i]$ {число пар с участием i-й частицы}
4. **for** $p = 0$ to $P_i - 1$ **do**
5. $df \Leftarrow F[i][p]$
6. $f_i \leftarrow f_i + df$
7. **end for**
8. Output: f_i

Переменная P_i используется для счёта частиц парных по отношению к i-й (для одной из потенциальных функций). Результирующая сила для i-й частицы (f_i) рассчитывается суммированием всех сил, вычисленных до этого ($F[i][p]$, строка 12–13). Эта часть программы может быть встроена в интегрирующее ядро для сокращения количества вычислительных ядер. На ГП новой архитектуры Fermi (от NVIDIA) наличие атомарного добавления atomicAdd(…) для вещественных величин позволяет производить сложение вычисленных значений сил при выполнении ядра, вычисляющего силы. Это поможет сократить количество обращений к памяти, что привлечёт к повышению эффективности программной реализации.

***Списки Верле*:** При использовании подхода распараллеливания по парам взаимодействующих частиц создание списка Верле сводится к формированию вектора *pairs* из всех взаимодействующих пар частиц в рамках одного слагаемого потенциала. Формирование вектора на ГП усложняется тем, что точная позиция в списке, куда добавляется информация о новой взаимодействующей паре, изначально неизвестна. Одно из возможных решений этой проблемы – использование функции atomicAdd(…) для целочисленых величин из библиотеки CUDA Software Development Kit [32], которая позволяет складывать целые числа, хранящиеся в глобальной памяти ГП, избегая конфликтов обращения к памяти, даже в случае, если множество потоков одновременно работает с информацией, хранящейся по одному и тому же адресу памяти. Параллельная работа потоков, выделяющих взаимодействующие пары и последовательно записывающих их, может привести к формированию списка Верле, не отсортированного в соответствии с порядковыми номерами частиц. Это может привести к неэффективному использованию кэш-памяти при чтении координат и несоблюдению условий коалесинга. Упорядочить лист Верле можно, если после его составления применить сортировку по индексу частицы (на ГП или ЦП). Другой вариант заключается в вычислении расстояний между частицами на ГП, копировании их в DRAM память ЦП и последующем составлении нового списка Верле. В силу того что при использовании динамики Ланжевена вода представлена неявно, обновлять список Верле можно редко. Из-за этого использование даже не оптимизированного кода не приведёт к серьёзным задержкам в расчёте.

2.3. Случайная сила. Для динамики Ланжевена требуется надёжный источник для получения 3N нормально распределённых псевдослучайных чисел $g_{i,\alpha}$ ($\alpha = x,y,z$), генерируемых на каждом шаге интегрирования, для расчёта трёх компонент вектора гауссовской случайной силы $G_{i,\alpha} = g_{i,\alpha}\sqrt{2k_BT\xi h}$, где k_B – постоянная Больцмана, T – температура системы, ξ – коэффициент трения, а h – временной интервал шага интегрирования. Гене-

ратор псевдослучайных чисел (ГПСЧ) создаёт ряд случайных чисел $u_{i,\alpha}$, равномерно распределённых на интервале [0,1]. Этот ряд, представляющий равномерные одинаково распределённые случайные величины, затем преобразуется в последовательность нормально распределённых псевдослучайных чисел с нулевым математическим ожиданием и дисперсией $g_{i,\alpha}$ с помощью преобразования Бокса-Мюллера (Box-Mueller) [48]. Несмотря на существование отдельных реализаций ГПСЧ на ГП, для динамики Ланжевена ГПСЧ должен быть встроен в ядро интегрирования с целью сокращения количества обращений чтения/записи к глобальной памяти ГП.

Самый простой подход к реализации ГПСЧ на ГП заключается в создании независимых генераторов для каждого потока (подход «один-ГПСЧ-на-поток»), что позволяет генерировать случайные числа «на лету» независимо в каждом потоке в процессе численного интегрирования уравнений Ланжевена. Для инициализации такого генератора ЦП создаёт N независимых наборов начальных состояний для N ГПСЧ и передаёт их в глобальную память ГП. Так как для генерации $3N$ нормально распределённых случайных величин $g_{i,\alpha}$ требуется $4N$ равномерных одинаково распределённых случайных величин $u_{i,\alpha}$, каждый поток далее считывает своё состояние и генерирует 4 нормально распределённых случайных числа для каждой из частиц. Затем ГПСЧ обновляет своё состояние в глобальной памяти ГП для того, чтобы использовать его на следующем шаге. Ещё один подход к реализации ГПСЧ основан на использовании одного состояния для всех вычислительных потоков ГП (подход «один-ГПСЧ-на-все-потоки»). Используя оба описанных подхода, мы разработали и протестировали ГП-реализации ГПСЧ, в основе которых лежат алгоритмы Hybrid Taus, Ran2, Фибоначчи с запаздываниями и вихрь Мерсенна. Нам удалось показать, что эти алгоритмы успешно проходят обязательные статистические тесты на случайность и генерируют наборы псевдослучайных чисел высокого качества.

2.4. Ядро интегрирования. На ГП уравнения движения Ланжевена могут быть решены одновременно для всех N частиц в N потоках, работающих параллельно. При использовании подхода распараллеливания по частицам процедуры расчёта силы могут быть встроены в ядро интегрирования. Это позволяет разработчику пользоваться переменными, содержащими значение координат частиц и хранящимися локально, считывая их всего один раз в начале вычислительного процесса и передавая каждой последующей процедуре. Однако данный подход существенно усложняет вычислительные ядра и увеличивает требования каждого потока к ресурсам, ограничивая число потоков, одновременно выполняющихся на каждом мультипроцессоре. Поэтому объединение всех вычислительных процедур в одно ядро эффективно только для маленьких систем, когда количество частиц соизмеримо с количеством вычислительных ядер на ГП. Так как все взаимодействия носят в той или иной степени локальный характер, текстурная кэш-память может быть эффективно использована для обеспечения доступа к координатам, хранящимся в глобальной памяти ГП. При использовании подхода распараллеливания по взаимодействующим парам частиц расчёт сил должен выполняться отдельным ядром, а суммирование всех сил (ядро суммирования, Алгоритм 5) встраивается в ядро интегрирования. Использование одного ядра для суммирования сил и численного интегрирования минимизирует количество вызовов ядер и таким образом экономит время, которое иначе бы тратилось на переключения между контекстами на ГП.

***Алгоритм 6*:** Численное интегрирование уравнений движения Ланжевена

1. $i \leftarrow$ индекс потока вычислений на ГП {равен индексу частицы}
2. $vecr_i <= r[i](t_n)$ {чтение координат i-й частицы в начале каждого шага}
3. $f_i \leftarrow \sum_v f_{i,v}$ {результирующая сила, действующая на i-ю частицу под действием нескольких потенциалов}
4. $g_i \leftarrow (g_x, g_y, g_z)$ {трёхмерный вектор с тремя нормально распределёнными случайными числами}
5. $r_i \leftarrow r_i + f_i h/\xi + g_i \sqrt{2k_BTh/\xi}$ {интегрирование 1-го порядка}
6. $r_i \Rightarrow r[i](t_{n+1})$ {сохранение координат в глобальной памяти в конце каждого шага}

Каждый поток вычисляет вектор координат лишь для одной частицы. Координаты частицы получены из глобальной памяти ГП (строка 2), сила, действующая на частицу (строка 3), а также безразмерный вектор случайной силы (строка 4) используются для вычисления координаты частицы на следующем временном шаге (строка 5). Соответствующие значения сил $f_{i,v}$, где индекс v указывает на различные слагаемые, входящие в потенциал, вычисляются с использованием подхода распараллеливания по частицам или по парам взаимодействующих частиц. Вычисленные новые позиции частицы сохраняются в глобальной памяти ГП (строка 6) для их использования на следующем временном шаге.

3. Программа SOP-GPU для моделирования динамики Ланжевена

3.1. Модель самоорганизующегося полимера (SOP). Мы использовали методологию реализации динамики Ланжевена на ГП для разработки CUDA-программы, выполняющей биомолекулярное моделирование на ГП. Молекулярное силовое поле представлено моделью самоорганизующегося полимера (SOP) – программа SOP-GPU [44]. Более ранние исследования показали, что модель SOP хорошо описывает механические свойства белков, в том числе зелёного флуоресцентного белка [54] и димера тубулина [55]. В модели SOP каждая частица описывается единым центром взаимодействия (атом C_α). Потенциальная энергия состояния белка V зависит от координат $\{r\} = r_1, r_2, \ldots, r_N$ частиц и определяется следующей формулой:

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

В уравнении (1) первое слагаемое описывает ковалентные связи в цепи при помощи конечно-растяжимого нелинейного эластичного потенциала (FENE) V_{FENE}. Расстояние между соседними частицами i и i+1 задаётся как $r_{i,i+1}$, а $r_{i,i+1}{}^0$ – его равновесное значение, R_0 = 2Å – индекс чувствительности к изменениям ковалентной связи. Потенциал Леннарда-Джонса ($V_{NB}{}^{ATT}$) был использован для описания нековалентных связей, фиксирующих нативное состояние (второе слагаемое). Если частицы i и j ($|i - j| > 2$), несвязанные ковалентными связями, расположены достаточно близко (в пределах R = 8Å) друг от друга в нативной структуре, то $\Delta_{ij} = 1$, и $\Delta_{ij} = 0$ – иначе. Мы использовали постоянное

значение ε_n = 1.5 *ккал/моль*, которое характеризует силу нековалентных связей. Все прочие взаимодействия описываются с помощью потенциала V_{NB}^{REP} и представляют собой силы отталкивания. Потенциал сгибания представлен в модели отталкиванием между частицами *i*, *i*+1 и *i*+2 с параметрами ε_r = 1 *ккал/моль* и $\sigma_{i,i+2}$ = 3.8 Å, которые определяют силу и радиус отталкивания. Последнее слагаемое введено, чтобы избежать самопересечений цепи и параметр σ равен 3.8 Å.

3.2. Тестовые расчеты. Для того чтобы протестировать работу программы SOP-GPU, мы осуществили тестовое моделирование процесса механического развёртывания домена *WW* человеческого белка *Pin*1 (PDB код 1PIN, табл.1). Данный белок был выбран для тестирования по двум причинам. Во-первых, развёртывание и динамика домена *WW* представляет практический интерес, и несколько исследовательских групп приложили значительные усилия для описания биохимических и биофизических свойств этого белка [29, 56, 57]. Во-вторых, домен *WW* – наименьший из независимо свёртывающихся доменов, полностью состоящих из β-листов, который был изучен экспериментально с применением методов атомной силовой спектроскопии [58,59]. По этим причинам домен *WW* широко используется при теоретическом изучении процессов сворачивания белков.

При проведении тестовых расчетов мы приняли во внимание следующие возможные источники ошибок: (1) ошибка точности вычислений, возникающая из-за разницы между использованием вещественной логики с одинарной точностью (на ГП) и с двойной точностью (на ЦП), (2) ошибки чтения/записи в общую память ГП (аппаратные ошибки), (3) точность программы SOP-GPU, то есть ошибки при выполнении численных операций (программные ошибки). Мы сравнивали результаты, полученные при моделировании с помощью нашей реализации программы SOP-GPU на ГП NVIDIA GeForce GTX 295 с результатами моделирования на двухпроцесорном Quad Core Xeon 2.66 *ГГц*, полученными при помощи проверенной программной реализации модели SOP на ЦП. С целью получить прямое сравнение скорости работы программы на ЦП и ГП, мы выбрали процессор и графическую карту одного технологического уровня. Сравнительные данные были получены при использовании одного ГП и одного ядра ЦП. Для моделирования динамики уравнения движения Ланжевена для каждой частицы r_i численно интегрировались с использованием схемы первого порядка точности (с шагом интегрирования *h*) [60],

$$r_i(t+h) = r_i(t) + f(r_i(t))\xi h + G_i(t), \qquad (2)$$

где $G_i(t)$ – случайная сила, а $f(r_i(t)) = -\dfrac{\partial V(r_i)}{\partial r_i}$ – результирующая сила ковалентных и нековалентных воздействий для *i*-й частицы (Уравнение (1)). Сравнительное моделирование механического развёртывания домена *WW* выполнялось при комнатной температуре (k_BT = 4.14 *пН/нм*) на протяжении 4×10^8 временных шагов длиной *h* = 20 *пс* и с применением стандартного значения вязкости воды ($\xi = 7.0\times10^5$ *пНпс/нм*). В каждой траектории мы закрепляли *N*-терминальный конец белковой цепи и прилагали к С-терминальному концу цепи домена *WW* силу, зависящую от времени $f_{ext}(t) = r_f t$, действующую в направлении вектора, соединяющего начальный и конечный элементы домена. Использовалось значение коэффициента силовой нагрузки $r_f = \kappa v_0$, где κ = 35 *пН/нм* – коэффициент упругости зонда, а v_0 = 2.5 *мкм/с* – скорость перемещения зонда.

Таблица 1. Количество кислотных остатков, ковалентных связей, нативных контактов, стабилизирующих свёрнутое состояние белка, и пар для дальнодействующего потенциала для ряда тестовых белков (домены *WW*, *Ig*27, *C*2*A*, фрагмент γ*C* и β*C* фибриногена, *D*-димер фибриногена), белковых волокон (мономер *Fb* и димер $(Fb)_2$ фибриногена) и белковых комплексов (капсула вируса *HK*97).

Белок	*WW*[1]	*Ig*27[2]	*C*2*A*[3]	γ*C*[4]	*D*-димер[5]	*Fb*[6]	$(Fb)_2$[7]	*HK*97[8]
PDB-код	1PIN	1TIT	2R83[9]	1M1J[10]	1FZB	3GHG	3GHG	1FT1[11]
Аминокислоты	34	89	126	517	1062	1913	3849	115140
Ковалентные	33	88	125	521	1072	1932	3839	114720
Нативные	65	255	328	1770	3498	5709	12560	467904
Пары	463	3573	7422	131101	558833	1821212	7389077	16178028[12]

Примечание:
[1] Домен *WW*, целиком состоящий из β-листов.
[2] Домен *Ig*27 титина человека.
[3] Домен *C*2*A* синаптотагмина человека *Syt*1.
[4] β*C* и γ*C* фрагменты фибриногена человека (*D*-домен).
[5] *D*-димер фибриногена человека (интерфейс *D*-*D*).
[6] Мономер фибриногена человека.
[7] Димер фибриногена человека, синтезированый из двух мономеров (3GHG) и *D*-димера (1FZB).
[8] Капсула вируса *HK*97 в состоянии Head II.
[9] Только *C*2*A* домен из структуры.
[10] β*C* и γ*C* фрагменты, начиная с цистеинового кольца *D*-домена.
[11] PDB код структурной единицы (капсомера), полная капсула доступна в базе данных ViperDB [64].
[12] Внутри радиуса обрезки в 200 Å.

Результаты вычислений, проведённых на ЦП и ГП, представлены на рис.2, где сравниваются кривые зависимости силы $f(R)$ от смещения зонда R, средняя температура $\langle T\rangle$, а также распределения сил денатурации, $p(f^*)$, построенные по 260 траекториям, рассчитанным на ЦП, и 300 траекториям, рассчитанным на ГП. Постоянная температуры ($d\langle T\rangle/dt$), механическая работа, выполняемая над системой ($w=\int_{R_0}^{R_{fin}} f(R)dR$), и распределение пиковых значений силы (f^*) являются точными физическими величинами, которые могут быть вычислены. Не принимая во внимание небольшие отклонения, возникающие из-за различных начальных условий, профили параметров $f(R)$ и $\langle T\rangle$, полученные на ЦП, близки к значениям этих же параметров, полученных на ГП. Небольшое падение для $\langle T\rangle$ вызвано началом развёртывания домена *WW* в момент $t \approx 0.15$ *мс*. Распределения сил развёртывания, рассчитанные и на ЦП, и на ГП, дают примерно одинаковые значения математического ожидания силы ($\langle f^*\rangle \approx 120.56$ *nH* на ЦП и $\langle f^*\rangle \approx 120.94$ *nH* на ГП), но незначительно отличающиеся значения стандартного отклонения ($\sigma_{f^*} \approx 5.83$ *nH* на ЦП и $\sigma_{f^*} \approx 6.58$ *nH* на ГП из-за маленького размера выборки – рис.2). Величина критической силы для процесса механического развёртывания находится в интервале 60 *nH* – 200 *nH*, что наблюдается для большого количества белков, состоящих из β-листов [58,61].

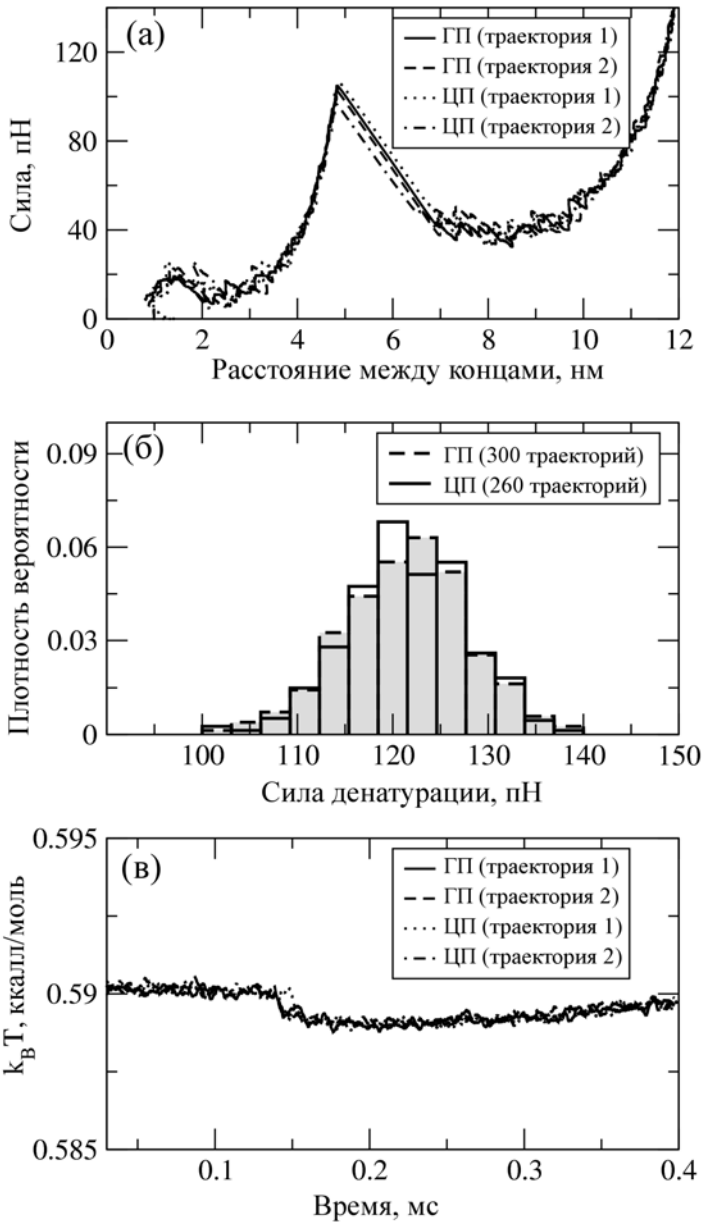

Рис.2. Сравнение результатов симуляций механической денатурации домена *WW*, полученных на ЦП и ГП. Панель (а): Примеры зависимости механического напряжения, испытываемого цепью белка, от растяжения белка, после применения метода скользящего среднего по 500 точкам. Панель (б): Гистограмма распределения сил денатурации $p(f^*)$, т.е. максимальных сил f^*, полученных из графиков зависимости силы от молекулярного растяжения. Гистограмма была построена с использованием интервала $h_{f^*} \approx 3.6$ *nH*. Панель (в): Примеры зависимостей средней температуры белка от времени, $<T(t)>$ (в единицах $k_B T$), соответствующих графикам зависимости силы от растяжения. К графикам был применён метод скользящего среднего по 500 точкам.

3.3. Точность численного интегрирования. Полная реализация моделирования динамики Ланжевена на ГП позволила получить длинные траектории белков и проследить их динамику на протяжении 10^9–10^{10} временных шагов. Как следствие, возникает вопрос о численной точности применяемой схемы интегрирования. Для моделирования динамики Ланжевена уравнения движения решаются с использованием схемы первого порядка точности (Уравнение (2)) [60]. Однако величина численной ошибки, которая может накапливаться в результате миллиардов итераций, неизвестна. Мы оценили численную точность интегрирования, рассматривая процесс механической денатурации белка. Для сравнения результатов моделирования растяжения белка $\Delta X(t) = X(t) - X_0$ с теоретическими данными была использована аналитическая модель броуновской частицы $X(t)$ в одномерном гармоническом потенциале $V(X) = K_{sp}\,(X - X_0)^2/2$, где X_0 – равновесное состояние, а K_{sp} – молекулярный коэффициент упругости [62].

В неравновесном состоянии под воздействием силы, меняющейся во времени, $f_{ext}(t) = r_f\, t$ – среднее положение частицы (расстояние от *C*-терминала до *N*-терминала) теоретически задаётся формулой:

$$\langle X(t)\rangle_{th} = X_0 e^{-t/\tau} + \frac{r_f \tau}{\xi}\left(t - \tau(1 - e^{-t/\tau})\right), \tag{3}$$

где $\tau = \xi/K_{sp}$ – параметр, характеризующий длительность протекания процесса. При моделировании растяжения среднее положение частицы на шаге n+1, $\langle X(t_{n+1})\rangle$, где $t_n = nh$, может определяться рекурсивно по положению частицы на предыдущем шаге n, $\langle X(t_n)\rangle$ с применением схемы интегрирования первого порядка:

$$\langle X(t_{n+1})\rangle_{sim} = \langle X(t_n)\rangle + \left(\frac{r_f t_n}{\xi} - \frac{\langle X(t_n)\rangle}{\tau}\right)h \tag{4}$$

или второго порядка:

$$\langle X(t_{n+1})\rangle_{sim} = \langle X(t_n)\rangle + \left(\frac{r_f t_n}{\xi} - \frac{\langle X(t_n)\rangle}{\tau}\right)h + \left(\frac{\langle X(t_n)\rangle}{2\tau^2} - \frac{r_f t_n}{2\tau\xi}\right)h^2 . \tag{5}$$

Таким образом, уравнения (3), (4) и (5) могут быть использованы для оценки точности численного интегрирования.

Расчёт $\langle \Delta X(t)\rangle$ выполнялся при комнатной температуре на протяжении $n = 10^9$ итераций из исходного равновесного состояния $X_0 = 0$, которое соответствует начальному молекулярному растяжению $\Delta X(0) = 0$, а $K_{sp} = 40$*nH/нм* находится в диапазоне 20–50 *nH/нм* значений, наблюдаемых в экспериментальных кривых силового растяжения белков. Использовались временные шаги длительностью $h = 1.25$ или 50 *nc*, а также параметры $\kappa = 10$ *nH/нм* и $v_0 = 1$ *мкм/с*, которые преобразуются в $r_f = 10^{-5}$ *nH/нс*. Это типичные значения коэффициента упругости зонда и скорости перемещения зонда (молекулярного растяжения), которые применяются в атомной силовой микроскопии. Был выбран коэффициент диффузии $D = k_B T/\xi = 1.5\times10^{-11}$ *см*^{2}*/с*, который соответствует растяжению ~ 150 *нм* в фибриногене *Fb*, полученному при силовой нагрузке на временном

интервале $t = 0.1\ c$ [21]. Медленную «диффузию молекулярного растяжения», вызванную силовым воздействием, $\Delta X(t)$ и описываемую моделью броуновской частицы с $D \approx 10^{-11}\ см^2/c$, следует отличать от свободной броуновской диффузии молекул белка в водном растворе, для которой D находится в диапазоне от $10^{-6}\ см^2/c$ до $10^{-8}\ см^2/c$.

Средние значения растяжения $\langle \Delta X(t_{n+1}) \rangle_{sim}$, полученные с применением схемы первого порядка (Уравнение 4) и схемы второго порядка (Уравнение 5), очень хорошо согласуются между собой и с теоретически рассчитанной кривой $\langle \Delta X(t) \rangle_{th}$ (Уравнение 3) для всех значений h. Полученные точечные значения $\langle \Delta X(t_{n+1}) \rangle_{sim}$ практически на всём интервале времени совпадают с теоретически рассчитанной кривой. Для количественной оценки точности использовалась относительная ошибка молекулярного растяжения, $\left| \langle \Delta X(t_{n+1}) \rangle_{sim} - \langle \Delta X(t) \rangle_{th} \right| / \langle \Delta X(t) \rangle_{th}$, накопленная к концу каждой из траекторий и усреднённая для 10^5 траекторий. Относительная ошибка оказалась меньше, чем 2×10^{-5} (1×10^{-5}) в случае схемы первого (второго) порядка, что значительно меньше, чем уровень 10^{-3}, который считается максимально приемлемым для относительной ошибки биомолекулярного моделирования. Эти результаты показывают, что в стохастических условиях водной бани (под действием случайных сил), ошибки численного интегрирования, возникающие при вычислении координат под влиянием механического воздействия, минимальны. Использование чисел с одинарной точностью вполне подходит для выполнения численного моделирования.

Таким образом, в контексте длительного моделирования поведения биомолекул на ГП, схема интегрирования первого порядка (алгоритм Ermac-McCammon [60]) может использоваться для достаточно точного описания их механических свойств, возникающих при силовых воздействиях в условиях, близких к физиологическим.

3.4. Измерение производительности. Сравнивалась итоговая производительность разработанного приложения для ГП (программа SOP-GPU) с оптимизированной версией реализации того же приложения для ЦП. Оба приложения описывают динамику Ланжевена молекулярной системы для домена *WW* в состоянии равновесия (рис.3, табл.1). Для полного использования ресурсов ГП мы представили вычислительную производительность программы SOP-GPU как функцию от числа независимых траекторий, обсчитываемых одновременно на одном ГП (подход «много-траекторий-на-одном-ГП»). Альтернативный этому вариант заключался в расчете одной траектории на одном ГП, но для систем разного размера (подход «одна-траектория-на-одном-ГП»). Результаты показывают, что для небольшой системы из 34 аминокислотных остатков (домен *WW*, Таблица 1), использование одного ГП позволяет ускорить вычисления при получении трёх и более независимых траекторий (точка перелома, рис.3а), т.к. небольшие системы не могут полностью загрузить все параллельные вычислительные ресурсы ГП. Эквивалентную загрузку одного ГП обеспечивает запуск обсчёта одной траектории для системы из $\sim10^2$ частиц (например, домен *Ig*27 или *C*2*A*, табл.1). Отметим, что время моделирования на ЦП линейно возрастает с увеличением количества запускаемых потоков, нагрузка на ГП в таком режиме остаётся почти линейной (примерно постоянной) вплоть до ~500 траекторий. Далее, ГП демонстрирует значительное превосходство в производительности над ЦП, достигая максимального ускорения в 80-90 раз (рис.3б). Моделирование

выполнялось достаточно долго для получения сходящейся оценки коэффициента ускорения ($n = 10^6$ шагов по $h = 40$ *псек* каждый). После этой точки ГП становится полностью загруженным, и время выполнения начинает линейно зависеть от количества запускаемых потоков (как на ЦП).

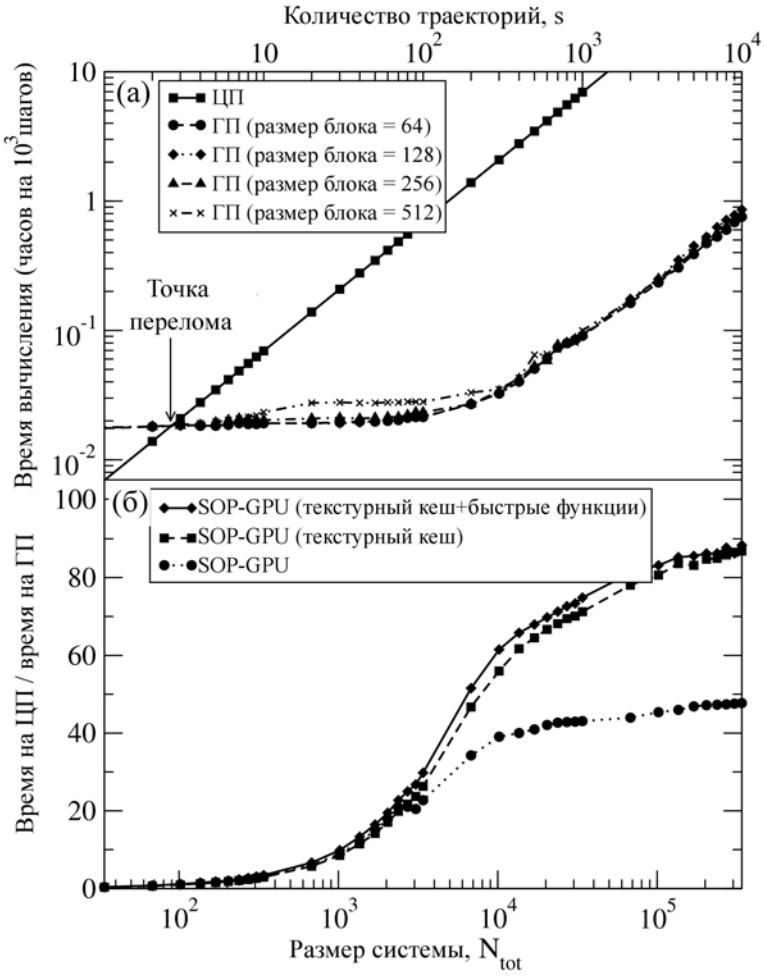

Рис.3. Панель (а): График зависимости времени вычисления 10^3 шагов интегрирования на ЦП и ГП от размера системы, N_{tot} в подходе «одна-траектория-на-одном-ГП» или от количества независимых траекторий, симулируемых одновременно, s в подходе «много-траекторий-на-одном-ГП». Для наглядности график показан с логарифмическим масштабом на обеих осях. Относительная производительность при расчете на ГП тестировалась с размерами блоков B=64, 128, 256 и 512. В то время как на ЦП симулируется только одна траектория, ГП может симулировать сразу много траекторий одновременно. Панель (б): Относительная производительность, или отношение скорости вычислений на ГП к скорости вычислений на ЦП (ускорение), полученное для приложения SOP-GPU как функция размера системы N_{tot} или количества траекторий s. Также показаны графики ускорения при использовании кэширования текстур и аппаратного ускорения математических функций.

Итоговое число потоков выполнения $M_{th} = m_B B$ и определяемое по количеству потоковых блоков m_B размера B примерно совпадает с произведением размера системы N на количество одновременно обрабатываемых траекторий s, то есть $M_{th} = Ns$. Так как заранее невозможно предсказать, какой размер блока B обеспечит лучшую производительность, были выполнены сравнительные запуски для $B = 64$, 128, 256 и 512. Полученные результаты показывают, что для достижения пиковой производительности, M_{th} должно превышать количество АЛУ на одном ГП в 20–40 раз. Например, на графической карте с 240 АЛУ (GeForce GTX 280 или GTX 295, или Tesla C1060) $M_{th} \approx 5\,000$-10 000. Это означает, что для домена *WW* ($N = 34$) оптимальное количество потоков $s \approx 300$ (рис.3). Таким образом, использование блоков небольшого размера более выгодно, когда $M_{th} \leq 3\,000$, например, при моделировании одной траектории для системы размера мономера фибриногена *Fb* ($N \approx 2\,000$), или 50-траекторий для системы из $N \approx 100$ частиц (домен *C*2*A*). Большие блоки можно использовать, если $M_{th} > 3\,000$ для выполнения моделирования нескольких траекторий для больших систем (димер фибриногена $(Fb)_2$, $N = 3\,849$) или одной траектории для очень большой системы (капсулы вируса *HK*97, $N = 115\,140$ – табл.1).

Для анализа затрат вычислительного времени и загрузки памяти на выполнение программы SOP-GPU были выполнены сравнительные запуски программы, моделирующей поведение нескольких разных белков и белковых образований (табл.1) с использованием подхода «одна-траектория-на-одном-ГП». Для построения зависимости времени выполнения итоговой версии программы от количества частиц моделируемой системы был использован стандартный инструмент профилирования, входящий в оболочку CUDA. Для всех тестовых систем было выполнено моделирование на промежутке в $n = 10^6$ временных шагов длительностью $h = 40$ *nc*, при размере блока $B = 64$ потока (рис.4). Для систем размером менее $N_{tot} < 4000$ аминокислотных остатков время моделирования остаётся примерно одинаковым, то есть составляет 10^7 шагов за час работы программы. Это неудивительно, так как $M_{th} \approx 5\,000$ – это приблизительное число потоков, необходимых для того, чтобы полностью загрузить ГП. Для систем, содержащих более 10^4 аминокислот, скорость вычислений падает линейно с увеличением размера системы N_{tot} (рис.4). Небольшие колебания скорости связаны с различиями в топологии тестовых систем, то есть количества ковалентных связей и нековалентных взаимодействий в нативной структуре белков (табл.1). Мы также показали, что объём памяти в современных графических картах – 1 *Гб* (GeForce GTX серии 200) и 4 *Гб* (Tesla C1060) – достаточен для описания больших биомолекулярных систем размером ~ 10^4 аминокислот (рис.4). Для систем размером более 10^5 частиц необходимо применять более сложные алгоритмы, включающие в себя реорганизацию списков Верле и списков возможных пар.

Важным моментом в полученной реализации является эффективное использование текстурного кэша для хранения промежуточных значений координат частиц. В сравнении с оптимизированной реализацией модели SOP, работающей на ЦП, реализация на ГП без использования текстурного кэша работает в ~10–50 раз быстрее, в зависимости от размера системы (рис.3б). Текстурный кэш позволяет ускорить работу программы, достигая ускорения в ~ 85 раз по сравнению с ЦП. Дополнительное ускорение может быть получено при помощи аппаратно ускоренных математических функций на ГП, доводя ускорение до ~ 90 раз.

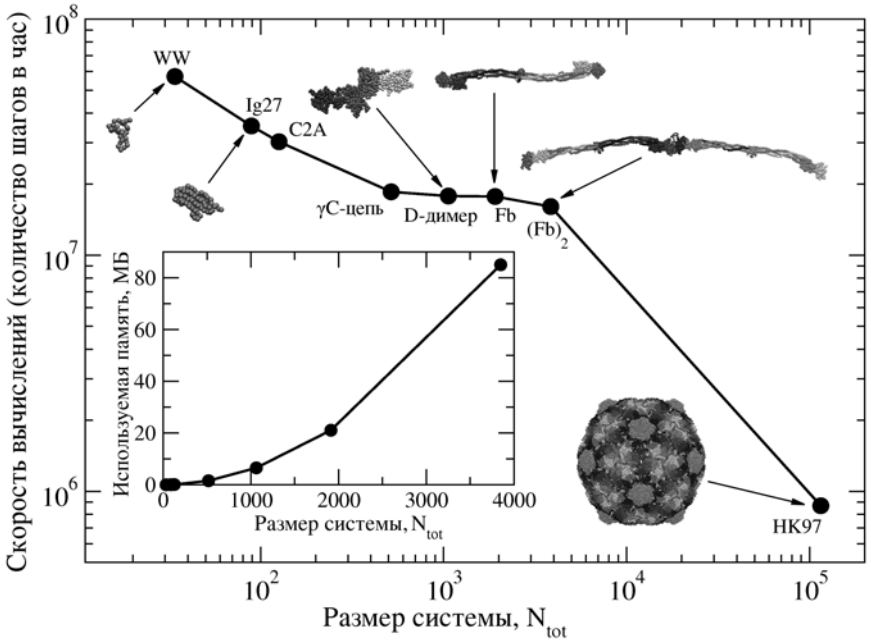

Рис.4. График зависимости скорости вычислений (количество шагов интегрирования, обсчитываемых за один час) от размера моделируемой системы (N_{tot}) для программы SOP-GPU. Для наглядности график показан в логарифмической шкале по обеим осям. Оценка производилась для ряда биомолекул, включая маленькие белки (домены *WW*, *Ig*27 и *C*2*A*), большие белковые фрагменты (цепь γ*C* и фрагмент «двойной-*D*» фибриногена), длинные нити фибрина (мономер *Fb* и димер $(Fb)_2$ фибриногена) и большую капсулу вируса *HK*97 (табл.1). На вставке показаны требования к памяти в зависимости от размера системы N_{tot}.

4. Заключение

В данной работе была разработана и протестирована, насколько нам известно, первая программная реализация биомолекулярных симуляций при помощи динамики Ланжевена, полностью работающая на ГП. Мы подробно описали два возможных подхода к реализации расчета парных взаимодействий: распараллеливание «по частицам» и «по взаимодействующим парам частиц», применяемых для расчета сил ковалентных и нековалентных взаимодействий между частицами. В работе использованы численные процедуры генерации псевдослучайных чисел с помощью алгоритма Hybrid Taus для описания случайных столкновений частиц биомолекул с молекулами растворителя (воды), описана программная реализация формирования списков Верле и численного интегрирования уравнений движения Ланжевена. Несмотря на то что в работе используется приближённая SOP-модель белка, в которой аминокислоты белка представлены в виде неделимых частиц (C_α-атомов) и функция потенциальной энергии содержит только парные взаимодействия между частицами, описанные методы могут применяться в сочетании с более сложными биомолекулярными силовыми полями, включая взаимодействия вида белок-

белок и белок-ДНК. Допустимо расширение модели для учёта других потенциалов, например, потенциал сгиба между тремя элементами аминокислоты, а также включение боковых цепей кислотных остатков. Ядро, выполняющее численное интегрирование, может быть преобразовано для описания термодинамики биомолекул с использованием динамики Ланжевена в незадемпфированном пределе.

Представленный формализм реализован в виде стандартного программного кода на CUDA, выполняющего биомолекулярное моделирование динамики Ланжевена (программа SOP-GPU). Сравнительные симуляции показали, что для тестовой системы *WW*-домена, результаты моделирования процесса денатурации под действием силы, выполненного на ГП, полностью согласуются с данными, полученными на ЦП (рис.2). В отдельной работе мы также сравнили результаты моделирования процесса развёртывания молекул человеческого белка фибриногена *Fb* (статья в процессе подготовки) и человеческого синаптотагмина *Syt*1 (статья в процессе подготовки) и обнаружили хорошую согласованность результатов вычислений на ГП и ЦП. На основании аналитической модели броуновской частицы в гармоническом потенциале мы оценили точность алгоритма численного интегрирования уравнений движения Ланжевена для описания механического растяжения белка. Было подтверждено, что разностная схема первого порядка обеспечивает достаточную точность расчётов даже на протяжении миллиардов шагов интегрирования.

ГП может использоваться для вычисления нескольких траекторий динамики Ланжевена для больших систем из $\sim 10^2$–10^3 частиц – подход «одна-траектория-на-одном-ГП». Это основной подход, используемый многими исследователями, работающими в данной области. Также было показано, как можно использовать ГП для получения большого количества независимых траекторий для небольших систем в $\sim 10^2$–10^3 аминокислот – подход «много-траекторий-на-одном-ГП». Этот подход может быть использован для получения статистически значимого количества траекторий для прямого сравнения функций распределения макропараметров систем, полученных экспериментально и теоретически. Важным моментом данной работы является эффективное использование текстурного кэша и применение аппаратно-ускоренных математических функций на ГП, которые позволяют ускорить работу программы в 90 раз по сравнению с реализацией на ЦП.

Полученные результаты подтверждают правильность работы программы SOP-GPU, которая может использоваться для описания механических свойств белков, нековалентных взаимодействий, стабилизирующих белки, белок-белковые комплексы и агрегаты, а также физических свойств крупных белковых образований. Комбинация упрощённой модели белка (SOP модель) и аппаратного ускорения при помощи современных ГП впервые позволяет производить симуляции больших белковых систем на биологически важных временных интервалах в 0.01–0.1 секунды за разумное время вычислений. При помощи данной программной реализации впервые становится возможным исследовать микромеханику процесса развёртывания волокон белка и изучать вязкоэластичные свойства биомолекулярных образований под воздействием экспериментальных и физиологических механических напряжений. Это позволяет теоретически интерпретировать кривые зависимости силы от растяжения, полученные экспериментально, с помощью методов динамической силовой спектроскопии, сокращая разрыв во временном диапа-

зоне между теоретическими и практическими исследованиями. Например, на ГП GeForce GTX 280 или GTX 295 требуется ~ 4 дня для расчёта одной траектории механической денатурации мономера фибриногена *Fb* с экспериментальной скоростью зонда 2.5 *мкм*/*с*. Для сравнения, расчёт одной траектории с помощью версии программы для ЦП занимает около 8 месяцев на процессоре Intel Core i7 2.66 *ГГц* с 6 *Гб* памяти. Требуется всего лишь ~ 20 дней для расчёта одной кривой механической индентации капсулы вируса *HK*97 с применением экспериментальной скорости движения зонда 2.5 *мкм*/*с*. На ЦП подобный расчет занял бы порядка пяти лет.

Из-за быстрого аппаратного развития ГП время моделирования на ГП значительно снизится вследствие запуска архитектуры Fermi (от NVIDIA), основанной на технологии MIMD (Multiple Instructions Multiple Data) [63]. Программа SOP-GPU, описанная в данной работе, после незначительных модификаций сможет выполняться на обновлённом аппаратном оборудовании с возросшими вычислительными ресурсами, что предположительно позволит проводить теоретические исследования ряда интересных научных проблем, для которых уже получены экспериментальные данные. Представленная в работе методология также может быть адаптирована для изучения других систем, таких как молекулярные моторы, нуклеосомы, липосомы и т.д.

СПИСОК ЛИТЕРАТУРЫ

1. *Stossel T.P., Condeelis J., Cooley L., Hartwig J.H., Noegel A., Schleicher M., Shapiro S.S.* Nat. Rev. Mol. Cell Biol. 2001, 2, 138–145.
2. *Johnson C.P., Tang H.Y., Carag C., Speicher D.W., Discher D.E.* Science 2007, 317, 663–666.
3. *Paul R., Heil P., Spatz J.P., Schwarz U.S.* Biophys. J. 2008, 94, 1470–1482.
4. *Leckband D.* Curr. Opin. Struct. Biol. 2004, 14, 523–530.
5. *McEver R.P.* Curr. Opin. Cell Biol. 2002, 14, 581–586.
6. *Marshall B.T., Long M., Piper J.W., Yago T., McEver R.P., Zhu C*. Nature 2003, 423, 190–193.
7. *Barsegov V., Thirumalai D.* Proc. Natl. Acad. Sci. USA 2005, 102, 1835–1839.
8. *Weisel J.W.* Biophys. Chem. 2004, 112, 267–276.
9. *Weisel J.W.* Science 2008, 320, 456–457.
10. *Lord S.T.* Curr. Opin. Hematol. 2007, 14, 236–241.10. Box G. E. P., Miller M. E. A note on the generation of normal random deviates // Ann. Math. Stat., 1958, Vol. 29, p. 610–611.
11. *Falvo M.R., Washburn S., Superfine R., Finch M., Brooks J.F.P., et al.* Biophys. J. 1997, 72, 1396–1403.
12. *Uetrecht C., Versluis C., Watts N.R., Roos W.H., Wuite G.J.L., et al*. Proc. Natl. Acad. Sci. USA 2008, 105, 9216–9220.
13. *Kuznetsov Y.G., Daijogo S., Zhou J., Semler B.L., McPherson A.* J. Mol. Biol. 2007, 347, 41–52.
14. *Kol N., Shi Y., Barlam D., Shneck R.Z., Kay M.S., et al.* Biophys. J. 2007, 92, 1777–1783.
15. *Ivanovska I.L., de Pablo P.J., Ibarra B., Sgalari G., MacKintosh F.C., et al*. Proc. Natl. Acad. Sci. USA 2004, 101, 7600–7605.
16. *Ivanovska I., Wuite G., Joensson B., Evilevitch A*. Proc. Natl. Acad. Sci. USA 2007, 104, 9603–9608.
17. *Steven A.C., B. H.J., Cheng N., Trus B.L., Conway J.F*. Curr. Opin. Struct. Biol. 2005, 15, 227–236.
18. *Carrion-Vazquez M., Li H., Lu H., Marszalek P.E., Oberhauser A.F., Fernandez J.M.* Nat. Struct. Biol. 2003, 10, 738–743.
19. *Schwaiger I., Sattler C., Hostetter D.R., Rief M*. Nature Mat. 2002, 1, 232–235.

20. *Brujic J., Hermans R.I., Walther K.A., Fernandez J.M.* Nature Phys. 2006, 2, 282–286.
21. *Brown A.E.X., Litvinov R.I., Discher D.E., Weisel J.W.* Biophys. J. 2007, 92, L39–L41.
22. *Smith D.E., Tans S.J., Smith S.B., Grimes S., Anderson D.L., Bustamante C.* Nature 2001, 413, 748–752.
23. *Roos W.H., Ivanovska I.L., Evilevitch A., Wuite G.* J. L. Cell. Mol. Life Sci. 2007, 64, 1484–1497.
24. *Brooks B.R., Bruccoleri R.E., Olafson B.D., States D.J., Swaminathan S., Karplus M.* J. Comp. Chem. 1983, 4, 187–217.
25. *Phillips J.C., Braun R., Wang W., Gumbart J., Tajkhorshid E., Villa E., Chipot C., Skeel R.D., Kalé L., Schulten K.* J. Comp. Chem. 2005, 26, 1781–1802.
26. *Berendsen H.J.C., van der Spoel D., van Drunen R.* Comp. Phys. Comm. 1995, 91, 43–56.
27. *Israelewitz B., Gao M., Schulten K.* Curr. Opin. Struct. Biol. 2001, 11, 224–230.
28. *Stone J.E., Phillips J.C., Freddolino P.L., Hardy D.J., Trabuco L.G., Schulten K.* J. Comp. Chem. 2007, 28, 2618–2640.
29. *Freddolino P.L., Liu F., Gruebele M., Schulten K.* Biophys. J. 2008, 94, L75–L77.
30. *Zink M., Grubmueller H.* Biophys. J. 2009, 96, 1767–1777.
31. ATI Stream Computing Technical Overview, AMD, 2009.
32. NVIDIA CUDA Programming Guide, version 2.3.1, NVIDIA, 2009.
33. NVIDIA CUDA C Programming Best Practices Guide, version 2.3, NVIDIA, 2009.
34. *Munshi A.* The OpenCL Specification, version 1.0, Khronos OpenCL Working Group, 2009.
35. *Rodrigues C.I., Hardy D.J., Stone J.E., Schulten K., Hwu W.-M.W.* GPU acceleration of cutoff pair potentials for molecular modeling applications. CF'08: Proceedings of the 5th conference on Computing frontiers, New York, NY, USA, 2008, pp 273–282.
36. *Phillips J.C., Stone J.E., Schulten K.* Adapting a message-driven parallel application to GPUaccelerated clusters. SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, Piscataway, NJ, USA, 2008, pp 1–9.
37. *Friedrichs M.S., Eastman P., Vaidyanathan V., Houston M., Legrand S., Beberg A.L., Ensign D.L., Bruins C. M., Pande V.S.* J. Comp. Chem. 2009, 30, 864–872.
38. *Davis J.E., 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, 2009, pp 176–186.
39. *van Meel J.A., Arnold A., Frenkel D., Zwart S.F.P., Belleman R.* Mol. Simul. 2008, 34, 259–266.
40. *Anderson J.A., Lorentz C.D., Travesset A.* J. Comp. Phys. 2008, 227, 5342–5359.
41. *Tozzini V.* Curr. Opin. Struct. Biol. 2005, 15, 144 – 150.
42. *Clementi C., Nymeyer H., Onuchic J.N.* J. Mol. Biol. 2000, 298, 937–953.
43. *Veitshans T., Klimov D., Thirumalai D.* Folding and Design 1997, 2, 1–22.
44. *Hyeon C., Dima R. I., Thirumalai D.* Structure 2006, 14, 1633–1645.
45. *Hyeon C., Onuchic J.N.* Proc. Natl. Acad. Sci. USA 2007, 104, 2175–2180.
46. *van der Spoel D., Lindahl E., Hess B., Kutzner C., van Buuren A.R., Apol E., Meulenhoff P.J., Tieleman D.P., Sijbers A.L.T.M., Feenstra K.A., van Drunen R., Berendsen H.J.C.* GROMACS User Manual, version 4.0, The GROMACS development team, 2009.
47. *Levesque D., Verlet L., Kürkijarvi J.* Phys. Rev. A 1973, 7, 1690–1700.
48. *Box G.E.P., Miller M.E.* Ann. Math. Stat. 1958, 29, 610–611.
49. GPU Gems 3, Nguyen, H., Ed., Addison-Wesley, 2008.
50. *Tausworthe R.C.* Math. of Comp. 1965, 19, 201–209.
51. *L'Ecuyer P.* Math. of Comp. 1996, 65, 203–213.
52. *Marsaglia G.* Published on sci.crypt.
53. *Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.* Numerical Recipes in C, 2nd ed., The Art of Scientific Computing, Cambridge University Press, 1992.

54. *M. Mickler R.I.D., Dietz H., Hyeon C., Thirumalai D., Rief M*. Proc. Natl. Acad. Sci. USA 2007, 104, 20268–20273.
55. *Dima R.I., Joshi H*. Proc. Natl. Acad. Sci. USA 2008, 105, 15743–15748.
56. *Ferguson N., Johnson C.M., Macias M., Oschkinat H., Fersht A.R*. Proc. Natl. Acad. Sci. USA 2001, 98, 13002–13007.
57. *Karanicolas J., III C.L.B*. Proc. Natl. Acad. Sci. USA 2003, 100, 3954–3959.
58. *Rief M., Gautel M., Oesterhelt F., Fernandez J., Gaub H*. Science 1997, 276, 1109–1112.
59. *Dietz H., Rief M*. Proc. Natl. Acad. Sci. USA 2004, 101, 16192–16197.
60. *Ermak D.L., McCammon J.A*. J. Chem. Phys. 1978, 69, 1352–1360.
61. *Carrion-Vazquez M., Oberhauser A.F., Fisher T.E., Marszalek P.E., Li H., Fernandez J.M*. Prog. Biophys. Mol. Biol. 2000, 74, 63–91.
62. *Doi M., Edwards S*. The Theory of Polymer Dynamics, International Series of Monographs on Physics, Oxford Science Publications, 1988.
63. NVIDIA'S Next generation CUDA Compute Architecture: Fermi, version 1.1, NVIDIA, 2009.
64. *Carrillo-Tripp M., Shepherd C.M., Borelli I.A., Venkataraman S., Lander G., Natarajan P., Johnson J.E., C.L. Brooks I., Reddy V.S*. Nucl. Acid. Res. 2009, 37, D436–D442.

Поступила в редакцию 29.03.2011.