2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

doi 10.26089/NumMet.v25r434 УДК 519.6

Применение параллельных вычислений при реализации метода выборки переходных поверхностей

С. В. Полуян
Государственный университет "Дубна",
Институт системного анализа и управления, Дубна, Российская Федерация
ORCID: 0009-0006-2296-6043, e-mail: poluyan@uni-dubna.ru

Н. М. Ершов
Московский государственный университет имени М. В. Ломоносова,
факультет вычислительной математики и кибернетики, Москва, Российская Федерация
ORCID: 0000-0001-5963-0419, e-mail: ershov@cs.msu.ru

Аннотация: Численная оценка константы скорости реакции является важной задачей в области биоинформатики, поскольку константа предоставляет информацию о кинетике связывания компонентов белкового комплекса. Один из подходов к выполнению данной оценки заключается в применении метода выборки переходных поверхностей, в основе которого лежит симуляция переходов между различными состояниями моделируемой системы по нескольким траекториям. Каждое состояние и переход характеризуются оценками энергии взаимодействия между компонентами белкового комплекса. В настоящей работе описаны схемы применения параллельных вычислений при построении множества траекторий. Приводятся принципы применения технологии MPI и библиотеки Intel oneTBB, а также демонстрируются результаты различных экспериментов. Целью исследования является определение наиболее эффективного инструмента параллельного программирования при построении множества траекторий в методе выборки переходных поверхностей в используемых вычислительных системах.

Ключевые слова: метод выборки переходных поверхностей, белок-белковые взаимодействия, константа скорости реакции, MPI, Intel oneTBB.

Для цитирования: Полуян С.В., Ершов Н.М. Применение параллельных вычислений при реализации метода выборки переходных поверхностей // Вычислительные методы и программирование. 2024. **25**, № 4. 453–463. doi 10.26089/NumMet.v25r434.

Using parallel computing in the transition interface sampling method implementation

Sergey V. Poluyan
Dubna State University,
Institute of System Analysis and Management, Dubna, Russia
ORCID: 0009-0006-2296-6043, e-mail: poluyan@uni-dubna.ru

Nikolay M. Ershov
Lomonosov Moscow State University,
Faculty of Computational Mathematics and Cybernetics, Moscow, Russia
ORCID: 0000-0001-5963-0419, e-mail: ershov@cs.msu.ru

Abstract: Numerical estimation of the reaction rate constant is an important task in the field of bioinformatics. This constant provides information on the binding kinetics of the protein complex

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

components. One approach to perform this estimation is to apply the transition interface sampling method, which is based on the simulation of transitions between different states of the modeled system along several trajectories. Each state and transition is characterized by the estimated energy of the interaction between the components of a protein complex. In this paper, we provide a way to apply parallel computing to multiple trajectory construction. We present the principles of using MPI technology and Intel oneTBB library, along with the demonstration of various experimental results. The purpose of the study is to determine the most effective parallel programming tool for constructing multiple trajectories in the transition interface sampling method within the given computational systems.

Keywords: transition interface sampling, protein-protein interaction, reaction rate constant, MPI, Intel oneTBB.

For citation: S. V. Poluyan and N. M. Ershov, "Using parallel computing in the transition interface sampling method implementation," Numerical Methods and Programming. **25** (4), 453–463 (2024). doi 10.26089/NumMet.v25r434.

1. Введение. В настоящей работе рассматриваются белковые комплексы вида белок–белок, где в роли двух компонентов комплекса выступают белки, являющиеся макромолекулами, образованными из полипептидных цепей. Процесс образования комплекса включает в себя один основной этап — связывание: процесс образования устойчивого комплекса при нековалентном взаимодействии друг с другом в растворе компонентов комплекса, а также соответствующее этой связи пространственное положение компонентов. При этом возможно рассматривать компоненты без каких-либо структурных изменений относительно свободного несвязного состояния в виде "твердых" тел [1], которые совершают в растворителе только поступательные и вращательные движения. Такое упрощение принято в настоящем исследовании. При этом для описания межмолекулярных взаимодействий и получения численной оценки энергии взаимодействия достаточно использовать оценочную функцию [2]. В оценочной функции для описания межмолекулярных взаимодействий используются классические эмпирические парные потенциалы Леннард-Джонса и Кулона, а энергия растворителя вычисляется в рамках модели неявного растворителя. Такие упрощения позволяют рассмотреть взаимодействия между компонентами на больших временны́х масштабах.

В соответствии с основным термодинамическим постулатом молекулярной биологии образование комплекса происходит потому, что комплекс является более термодинамически стабильным, чем свободные несвязанные компоненты [3]. В простейшем случае процесс связывания описывается моделью вида ключ–замок, которая представляется в виде $A + B \leftrightarrow AB$, где A и B являются компонентами комплекса.

Кинетика образования комплекса определяется через кинетику полуреакций. Образование и распад комплекса, как правило, записывают в виде $A + B(k_1) \rightarrow AB$ и $AB(k_2) \rightarrow A + B$, где k_1 — константа скорости прямой реакции, k_2 — константа скорости обратной реакции. Образование комплекса зависит от скорости, с которой он образуется при ассоциации реагентов [3] ($k_{\text{on}}[A][B]$, где k_{on} — константа скорости ассоциации), и скорости, с которой комплекс распадается ($k_{\text{off}}[AB]$, где k_{off} — константа скорости диссоциации). Приведенные выше величины определяют константы связывания.

Численная оценка константы скорости ассоциации k_{on} позволяет судить о кинетике процесса связывания, а оценка k_{on} для белковых комплексов различного вида является одной из основных задач для биохимиков, медицинских химиков и биоинформатиков [3]. Возможно выполнить оценку констант связывания компонентов белковых комплексов экспериментально. Например, с помощью систем поверхностного плазмонного резонанса или флуоресцентной спектроскопии [3]. Такие методы накладывают ряд ограничений, которые сопутствуют проведению физических экспериментов.

На сегодняшний день можно отметить ограниченное число исследований, в которых происходит оценка констант связывания посредством проведения вычислительных экспериментов, имеющих в качестве входных параметров пространственное представление комплексов. Например, в работе [4] с помощью метода молекулярной динамики были произведены оценки констант ассоциации. Очевидно, что применение метода молекулярной динамики требует значительных вычислительных затрат. Недостатком приведенного в исследовании способа является малый временно́й масштаб, несмотря на применение крупнозернистой модели белкового комплекса.

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

Помимо метода молекулярной динамики, исследователи применяют различные численные методы для оценки указанной константы. Например, метод TransComp [5], основанный на теории "переходного" комплекса (transient-complex theory) [6]. В методе рассматривается иная кинетическая модель вида $A + B \rightleftharpoons A^*B \rightarrow AB$, где A^*B — "переходный" комплекс. К недостаткам метода следует отнести наличие верхней и нижней границ оценок, которые заданы авторами в рамках разработанной теории.

Следующим примером метода для оценки константы ассоциации является кинетический метод Монте-Карло [7]. В работе демонстрируется применение метода с использованием крупнозернистой модели для нескольких комплексов вида белок–белок. Недостатком метода является необходимость использования референтного набора белковых комплексов с известными значениями констант связывания, без которых оценка константы связывания невозможна.

Рассматриваемый в настоящей работе метод выборки переходных поверхностей (multiple state transition interface sampling) [8] лишен вышеупомянутых недостатков и ранее использовался для оценки констант скорости в различных молекулярных системах. Важно отметить, что до настоящего времени не были разработаны конкретные программные инструменты для оценки константы на основе трехмерной структуры с использованием рассматриваемого метода.

Помимо перечисленных методов, существуют различные способы "ускорения" молекулярной динамики, позволяющие выполнить численную оценку. Однако среди существующих подходов только два вышеупомянутых метода были разработаны специально для оценки константы для комплексов вида белок–белок. Методы продемонстрировали положительную корреляцию с экспериментальными значениями на небольших наборах тестовых комплексов, каждый из которых формировался авторами по различным критериям. Следует отметить, что на данный момент для упомянутых методов не проводилось исследование, направленное на оценку степени корреляции с использованием независимого тестового набора комплексов.

Верифицировать работу рассматриваемого метода планируется на основе базы данных SKEMPI [9]. В ней для различных комплексов приводятся константы скорости ассоциации, а сама база сформирована на основе экспериментальных данных. В базе для каждого комплекса присутствует ссылка на работу (или на несколько работ) с результатами эксперимента. В базе данных SKEMPI более 1500 комплексов белок–белок с известной константой скорости ассоциации.

В качестве объекта исследования при проведении вычислительных экспериментов выступает белковый комплекс 2OOB (код в базе данных Protein Data Bank [10]), для которого известна константа связывания, а также численная оценка методом TransComp.

Реализация параллельной версии первого шага исследуемого метода численной оценки константы скорости производится на языке программирования C++. Предметом настоящего исследования являются средства синхронизации и многопоточности языка программирования C++ и высокоуровневые шаблоны библиотеки oneTBB [11]. Основной целью исследования является определение наиболее эффективного примитива синхронизации в используемых вычислительных системах:

- служебный сервер для параллельных вычислений, предоставляемый в рамках инфраструктуры университета "Дубна";
- учебно-тестовый полигон гетерогенной платформы HybriLIT [12] ОИЯИ, позволяющий выполнять вычисления на нескольких вычислительных узлах.

Платформа HybriLIT будет задействована на этапе исследования метода на обширном наборе тестовых комплексов, в то время как использование служебного сервера запланировано после создания программного инструмента, который позволит загружать комплекс через веб-сервер для выполнения оценки. В перечисленных системах присутствуют различия в программно-аппаратном окружении, что может повлиять на эффективность применения средств параллельного программирования.

Основным ожидаемым преимуществом искомого инструмента параллельного программирования является высокая эффективность. Для поиска данного инструмента в исследовании проводится сравнение реализаций одного из паттернов параллельного программирования и демонстрируются результаты экспериментов, которые выполнены в различных вычислительных средах. Практическая значимость заключается в возможности определения оптимальных стратегий параллельной реализации для конкретных условий использования и типа решаемой задачи.

2. Постановка задачи. Связывание компонентов белкового комплекса можно рассматривать в парадигме классической теории переходного состояния, где в процессе моделирования система через последовательность квазиравновесных состояний движется в сторону наименьшей полной энергии по пути

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

с наименьшими энергетическими барьерами. Метод выборки переходных поверхностей основан на идее генерации траекторий, которые позволяют исследовать редкие переходные события между различными состояниями системы. Первым шагом этого метода является выборка траекторий вдоль так называемых переходных путей (или “поверхностей”), которые соединяют различные начальные состояния с конечным. Каждый путь характеризуется одним или несколькими переходными состояниями, часть из которых для нескольких путей совпадает. Объединив все возможные состояния, возможно выполнить кластеризацию переходных состояний, а затем выделить основные траектории связывания. Используя выделенные траектории связывания, возможно выполнить численную оценку кинетических параметров связывания, в том числе выполнить оценку константы k_{on}, моделируя переходы между состояниями [8].

При генерации множества траекторий и при поиске переходных путей связывания необходимо многократно оценивать энергию взаимодействия между компонентами комплекса с использованием оценочной функции. Расположение компонентов комплекса на различных расстояниях оказывает влияние на время, затрачиваемое на оценку энергии. На рис. 1 представлены компоненты комплекса 2OOB. График на рис. 2 показывает зависимость времени выполнения оценки от расстояния между компонентами комплекса при смещении вдоль одной оси одного из компонентов комплекса.

Следует отметить, что генерируемые траектории и соответствующие найденные пути перехода между состояниями имеют разную “длину”. Каждый путь определяет множество различных расположений компонентов в пространстве относительно друг друга. При этом структурное представление компонентов значительно варьируется, что определяет неравномерность времени оценки энергии взаимодействия. На рис. 3 представлены различные пути связывания, соответствующие им профили и диаграмма размаха, которая демонстрирует значительное отклонение времени поиска при определении пути.

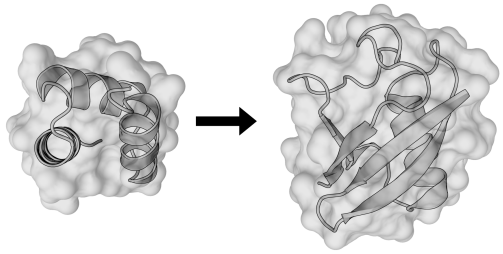

Рис. 1. Визуализация компонентов белка 2OOB и связывания

Fig. 1. Visualization of 2OOB protein components and binding

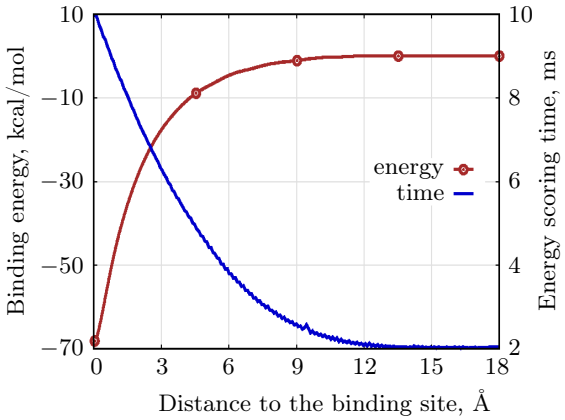

Рис. 2. Энергия связывания и время оценки энергии связывания в зависимости от расстояния между компонентами комплекса

Fig. 2. Binding energy and scoring time as functions of the distance between the components of the complex

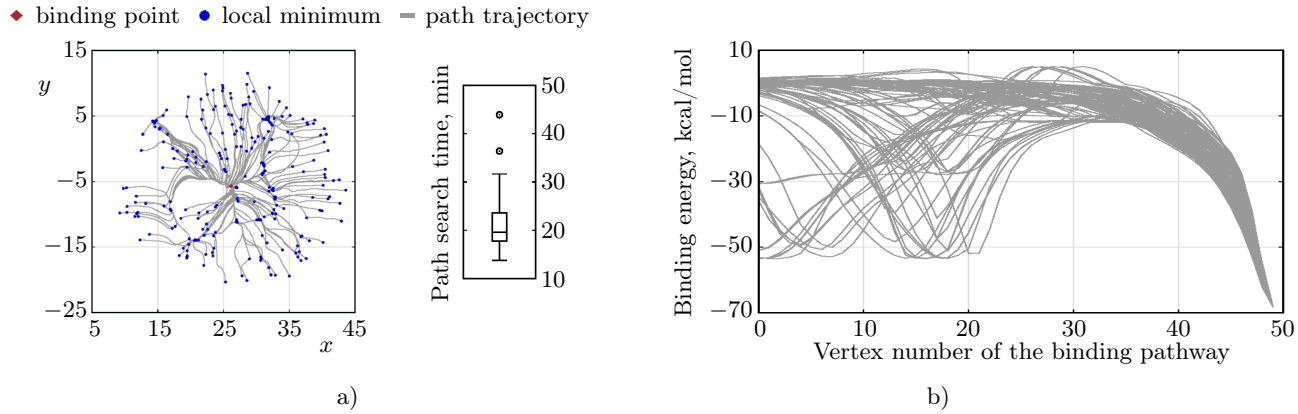

Рис. 3. Связывание компонентов белкового комплекса: a) пути связывания и диаграмма размаха; b) профили путей связывания

Fig. 3. Binding of components of the protein complex: a) binding pathways and box plot; b) binding pathway profiles

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

В исследовании поиск пути с наименьшими энергетическими барьерами производился с помощью метода струны (string method) [13]. В методе происходит итеративная эволюция точек, каждая из которых характеризует пространственное положение одного компонента комплекса. Точки перемещаются под действием градиентных сил до достижения критерия сходимости.

Очевидно, что разделение множества траекторий на блоки одинакового размера с параллельной обработкой каждого блока неэффективно. Задача заключается в организации параллельных вычислений на основе исходного множества траекторий с учетом значительного отклонения времени нахождения каждого пути. Следует отметить, что постановка задачи является в значительной мере стандартной и, в определенном смысле, тривиальной, однако она предусматривает возможность достижения максимальной эффективности параллельных вычислений.

Для решения задачи целесообразно использовать паттерн параллельного программирования “пул потоков” (thread pool). Помимо упомянутого паттерна при использовании системы с несколькими вычислительными узлами применялась парадигма параллельного программирования “ведущий–ведомый” (master–slave) [14]. Если рассматривать паттерн в упрощенном виде, задача делится на несколько подзадач, которые выполняются параллельно на различных вычислительных узлах с использованием паттерна “пул потоков”, а затем результаты этих подзадач объединяются.

3. Параллельная реализация. В исследовании реализация паттерна “пул потоков” производилась с использованием очереди задач. Такая реализация возможна с применением средств языка программирования C++ или шаблонов библиотеки oneTBB. Следует отметить, что возможно смоделировать задачу с помощью синтетического теста, результаты которого, как будет продемонстрировано ниже, не могут полностью указать на наиболее эффективный подход к реализации.

Рассматриваемый шаг метода может быть описан в упрощенной форме следующим образом. Каждая траектория характеризуется фиксированным набором точек, которые распределяются по определенному правилу в области связывания. Конечная точка каждой траектории определяет место связывания и одинакова для всех траекторий. Таким образом, имеется массив траекторий размером `paths_number`, в котором каждая траектория определяется уникальным `path_id`. Для каждой траектории требуется выполнить процедуру `calculate` поиска пути связывания, что сопряжено с существенными вычислительными затратами.

В результате выполненной работы произведена параллельная реализация, которая поддерживает вычисления на нескольких вычислительных узлах. Для организации распределенных вычислений использовалась технология MPI. Принцип распределения MPI процессов следующий: на одном вычислительном узле создается один MPI процесс, внутри которого запускаются вычислительные потоки, работающие в системе с общей памятью. Следует отметить, что временны́е расходы по передаче при обмене между вычислительными узлами минимизированы, поскольку каждая траектория передается в формате одномерного массива чисел с двойной точностью. Множество траекторий распределяется главным процессом на вычислительные узлы блоками фиксированного размера перед началом вычислений. Поскольку существенная часть траекторий исключается после окончания вычислений, в главный процесс возвращается существенно меньшее число путей, которые также передаются в формате одномерного массива.

На этапе реализации метода использовались следующие инструменты: Valgrind [15] для отладки с целью минимизации выделения динамической памяти и Intel Advisor [11] для анализа производительности в контексте параллельных вычислений.

В ходе исследования были рассмотрены три способа параллельной организации вычислений в рамках одного вычислительного узла, каждый из которых представлен ниже.

3.1. Блочная реализация. В качестве простейшего подхода к параллельной обработке массива траекторий использовалось блочное параллельное выполнение. Подход представлен для демонстрации необходимости и преимуществ применения паттерна “пул потоков”. Исходный массив траекторий разбивался на блоки фиксированного размера. Для этого использовался кратный числу потоков размер блоков. После этого производилась независимая друг от друга обработка каждого блока соответствующим потоком. Очевидно, что по вышеперечисленным в предыдущем разделе причинам такой подход приводит к неравномерной загрузке потоков и снижению эффективности параллельной обработки и был реализован для демонстрации преимуществ применения следующих способов.

Реализация блочного выполнения произведена на языке программирования C++ с использованием инструментария `std::thread`. Фрагмент реализации представлен в репозитории [16]. В начале создается массив потоков `threads` размером `n_threads`. Затем вычисляется размер каждого блока `part_size`.

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

В первом цикле создаются и запускаются потоки, каждый из которых использует собственный уникальный идентификатор потока `thread_id`, а также начальный и конечный индексы в массиве траекторий (`a` и `b`). В последнем цикле каждый поток ожидает завершения своего выполнения путем вызова `join()`. Этот механизм гарантирует, что главный поток приостанавливает свое выполнение до тех пор, пока все дочерние потоки не завершат свою работу.

3.2. Использование Intel oneTBB. Реализация паттерна “пул потоков” с использованием очереди и средств библиотеки oneTBB выполнена двумя способами. В обоих случаях создавался потокобезопасный контейнер `tbb::concurrent_queue`, в который добавлялись все уникальные идентификаторы траекторий `path_id` (индексы массива траекторий). В первом случае применялся высокоуровневый шаблон `tbb::parallel_for` как наиболее универсальный подход организации параллельных вычислений средствами библиотеки oneTBB. Во втором случае использовался шаблон `tbb::task_group`, предоставляемый библиотекой для управления задачами с различной продолжительностью выполнения. Фрагмент реализации представлен в репозитории [16].

3.3. Использование встроенных средств языка программирования C++. В настоящее время стандарт языка программирования C++ не включает в себя потокобезопасные контейнеры. Однако можно создать потокобезопасную очередь с использованием различных механизмов синхронизации, доступных в языке. Поскольку результаты синтетического теста [16] демонстрируют практически идентичное время выполнения различных примитивов синхронизации, в исследовании был выбран базовый механизм синхронизации — `std::mutex`. Фрагмент реализации, код теста и результаты представлены в репозитории [16]. Внутри цикла `while` создается объект типа `lock_guard`, который блокирует мьютекс при входе в блок. Это гарантирует, что только один поток имеет доступ к очереди в определенный момент времени. Дополнительный локальный блок внутри цикла используется для ограничения области видимости объекта и для того, чтобы мьютекс был разблокирован после выполнения операций, которые требуют защиты, так как объект `lock_guard` автоматически разблокирует мьютекс при выходе из своей области видимости. Такой подход реализует идиому RAII (Resource Acquisition Is Initialization), что гарантирует освобождение ресурсов автоматически при выходе из области видимости объекта.

4. Вычислительные эксперименты. Поскольку основной целью исследования является определение наиболее эффективного примитива синхронизации в используемых вычислительных системах, важно учитывать их особенности и конфигурацию при запуске экспериментов. В случае служебного сервера, предоставляемого в рамках инфраструктуры университета “Дубна”, использовался один процессор Intel Xeon E5-2650 с 8 физическими ядрами. При экспериментах на учебно-тестовом полигоне гетерогенной платформы HybriLIT вычисления производились на узлах, каждый из которых включает в себя два 12-ядерных процессора Intel Xeon E5-2695 v2.

Необходимо отметить, что использованные в экспериментах вычислительные узлы не поддерживают архитектуру NUMA. Естественно предположить, что наличие двух сокетов на вычислительном узле позволит использовать преимущества NUMA-архитектуры. Однако в случае некоторых узлов платформы HybriLIT это не так. На момент проведения исследования для рассматриваемой группы узлов, которые присутствуют в очереди задач для вычислений с названием “cpu”, использовался гипервизор, конфигурация которого исключает поддержку NUMA. Поскольку поддержка отключена на уровне операционной системы, запускаемые приложения не могут использовать NUMA-архитектуру.

При запуске устанавливалась переменная окружения `OMP_PLACES` для распределения потоков по процессорным ядрам. Поскольку поддержка NUMA-архитектуры на узлах отсутствует, необходимость в создании группы потоков и управлении их размещением отпадает, поэтому для переменной устанавливалось значение `cores`. Помимо этого, через систему управления задачами SLURM устанавливалась опция с указанием максимально допустимого количества процессорных ядер, чтобы на вычислительных узлах присутствовала только одна задача.

На каждой итерации в методе струны присутствует один времязатратный шаг — вычисление оценки энергии взаимодействия, который состоит из двух этапов: поиска взаимодействующих атомов и вычисления энергии взаимодействия с использованием эмпирических парных потенциалов. При поиске весомым фактором, ограничивающим производительность, является пропускная способность памяти, поскольку в простейшем случае предполагается перебор всех атомов комплекса. Поиск может быть значительно оптимизирован применением k-d-дерева, что, однако, никак не исключает упомянутый фактор, поскольку только сокращает количество обращений к памяти.

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

Несмотря на то что используемые потенциалы [2] в строгом смысле нелинейны и часть из них построена с применением элементарных функций, при вычислении требуется получение коэффициентов в соответствии с типом взаимодействующих атомов, которое также предполагает чтение памяти.

В результате применения Valgrind выявлено, что при поиске пути количество выделенной динамической памяти с течением времени остается постоянным, а наибольшее время выполнения занимают функции, определяющие смещение компонента и оценку энергии взаимодействия, в то время как наиболее часто вызываемая функция используется для получения координат атомов. Помимо этого, с помощью утилиты Perf определен приемлемый уровень кэш-промахов. Процентное отношение кэш-промахов к общему числу обращений для всех уровней кэша процессора составило 3.37%.

На первом этапе экспериментов проведен анализ производительности параллельной реализации с базовым механизмом синхронизации. На рис. 4 представлены результаты использования инструмента Intel Advisor при запуске эксперимента по поиску 72 путей на одном вычислительном узле кластера HybriLIT с использованием 24 потоков. На рис. 4 представлена диаграмма модели Cache-Aware Roofline [17], на которой отмечены полученные пропускные способности различных уровней кэша и памяти, а также пики производительности для различных типов вычислений. Метками обозначены наиболее нагруженные функции и циклы. Цвета и размеры отражают относительное время выполнения, т.е. показывают долю времени, затраченного на выполнение определенного “участка” программы (функции отмечены только цветом, циклы обведены), от общего времени выполнения программы.

Выделенная зеленой областью группа функций выполняется при вычислении парных потенциалов. Участки, которые выделены тремя метками циклов по центру диаграммы (желтым, красным и зеленым цветами), выполняются при обходе k-d-дерева и при вычислении расстояния между потенциально взаимодействующими атомами. Находящаяся в “scalar compute bound” области функция вызывается при изменении пространственного положения компонента комплекса.

Маркер в виде креста на диаграмме (рис. 4) означает совокупную производительность всей программы, включая все функции и циклы. Несмотря на то, что маркер находится выше линии пропускной способности памяти, производительность значительно ограничена пропускной способностью памяти. Используя метрики производительности Intel Advisor, можно сделать вывод, что текущая реализация ограничена как вычислительной мощностью, так и пропускной способностью памяти.

На втором этапе экспериментов использовался служебный кластер, состоящий из одного процессора Intel Xeon E5-2650 с 8 физическими ядрами. Эксперименты проводились с использованием 8 потоков без применения технологии MPI. Использовался компилятор Clang (версия 17) с предоставляемой стандартной библиотекой шаблонов (libc++). Результаты ускорения работы и эффективности блочной реализации (block), реализации с использованием библиотеки oneTBB (TBB pf, TBB tg) и реализации с использованием средств языка программирования C++ (mutex) представлены на рис. 5. Сравнение проводилось относительно последовательной версии программы, с помощью которой были найдены 120 путей,

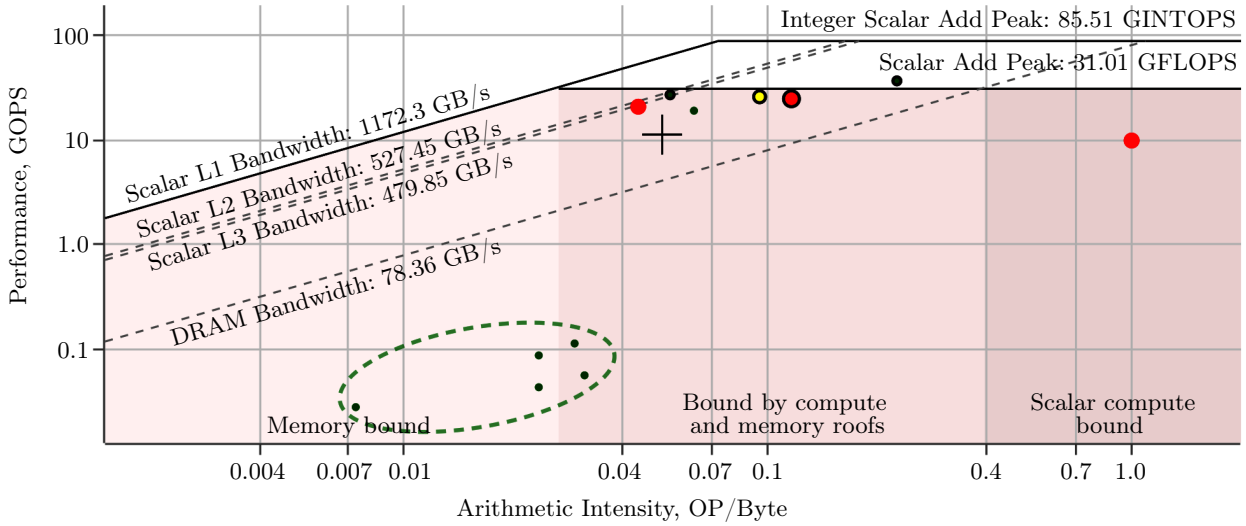

Рис. 4. Построенная средствами Intel Advisor диаграмма Roofline-модели

Fig. 4. A diagram Roofline model constructed using Intel Advisor

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

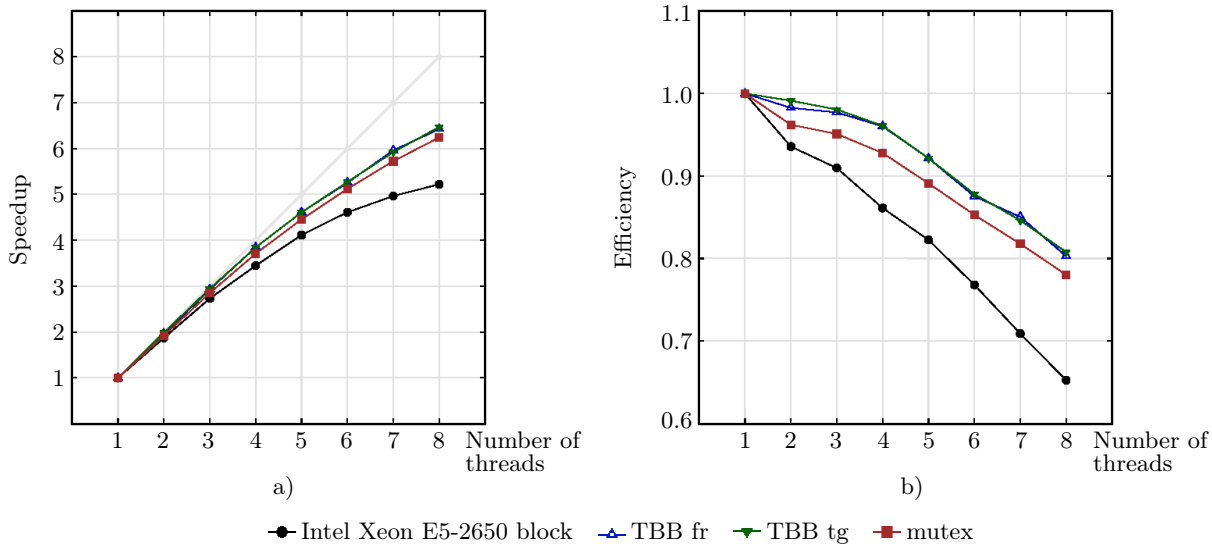

Рис. 5. Метрики вычислительного процесса на служебном сервере университета “Дубна”: a) ускорение; b) эффективность

Fig. 5. Metrics of computational process by using the Dubna State University service server: a) speedup; b) efficiency

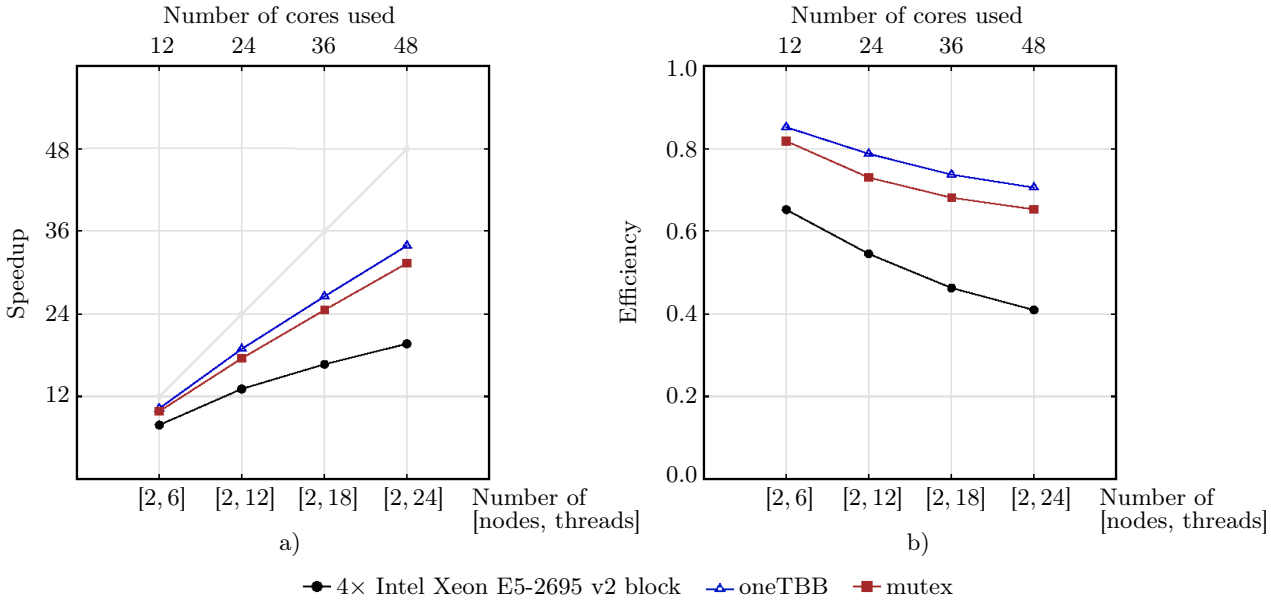

Рис. 6. Метрики вычислительного процесса на двух вычислительных узлах платформы HybriLIT: a) ускорение; b) эффективность

Fig. 6. Metrics of computational process by using two compute nodes of the HybriLIT platform: a) speedup; b) efficiency

изображенных на рис. 3. Выполнение последовательной версии программы заняло 20 часов. Единственным преимуществом последовательной версии является малая пространственная сложность, поскольку для оценки энергии взаимодействия на основе одного из компонентов комплекса (недвижимого) требуется однократное построение k-d-дерева для эффективного поиска взаимодействующих атомов. В случае многопоточной версии для каждого потока однократно создавалось дерево, т.е. использование памяти возрастает кратно числу потоков. Следует отметить, что предъявляемые методом требования к памяти незначительны, а сама память при поиске пути выделяется однократно.

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

На третьем этапе экспериментов использовались ресурсы учебно-тестового полигона гетерогенной платформы HybriLIT ОИЯИ. Использовался компилятор GCC (версия 12) с предоставляемой стандартной библиотекой шаблонов (libstdc++). Распределенные вычисления проводились с использованием технологии MPI двумя процессами, что отмечено на рис. 6. В экспериментах рассматривалось применение шаблона `tbb::parallel_for`.

Как видно из рис. 5, 6, во всех экспериментах наибольшую эффективность показала параллельная реализация с использованием библиотеки oneTBB. Причины большей эффективности связаны с механизмом планирования вычислений [18], который реализован в библиотеке oneTBB. Указанный механизм, использование которого происходит при вызове `tbb::parallel_for` и `tbb::task_group`, представляет собой автоматизированный способ управления выполнением потоков, который способствует оптимальному использованию ресурсов и повышению производительности вычислений, причем это наблюдается в обоих случаях. Следует отметить, что в экспериментах использовались разные компиляторы с различными реализациями стандартной библиотеки шаблонов, но с одинаковой версией библиотеки oneTBB (версия 2021.12).

Во всех экспериментах время вычисления последовательной версии ограничивалось 24 часами. Число обработанных за указанное время траекторий в дальнейшем использовалось при запуске параллельных версий.

5. Заключение. При использовании двух вычислительных узлов с распределенной памятью для параллельных вычислений с применением библиотеки oneTBB была достигнута эффективность приблизительно в 70%. Хотя полученная эффективность достаточно высока, ее можно охарактеризовать как недостаточную. Дальнейшее исследование будет посвящено оптимизации процесса формирования выборки с целью повышения эффективности, параллелизации следующих этапов метода выборки переходных поверхностей, а также изучению масштабируемости разработанной параллельной реализации.

В результате выполненной работы проведена параллельная реализация основного этапа метода выборки переходных поверхностей несколькими различными способами и достигнута основная цель исследования — выявлен наиболее эффективный способ организации параллельных вычислений для рассматриваемых вычислительных систем, который основан на применении библиотеки oneTBB. Следует отметить, что применение библиотеки oneTBB позволило увеличить эффективность вычислений примерно на 5% по сравнению с реализацией, осуществленной стандартными средствами языка программирования C++.

Для рассматриваемого белкового комплекса получен ряд численных оценок константы k_{on}, которые коррелируют с известными численными и экспериментальными оценками. Расширение тестового набора белковых комплексов и проведение вычислительных экспериментов для оценки констант у различных комплексов являются темами дальнейших исследований.

Список литературы

1. *Хрущев С.С., Абатурова А.М., Дьяконова А.Н. и др.* Моделирование белок-белковых взаимодействий с применением программного комплекса многочастичной броуновской динамики ProKSim // Компьютерные исследования и моделирование. 2013. **5**, № 1. 47–64. doi 10.20537/2076-7633-2013-5-1-47-64.
2. *Полуян С.В., Никулин Д.А., Ершов Н.М.* Разработка и верификация оценочной функции для учета межмолекулярных взаимодействий в белковых комплексах // Материалы Всероссийской конференции “Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем”, 17–21 апреля 2023. М.: Изд-во РУДН, 2023. 231–235. https://events.rudn.ru/event/198/attachments/550/1474/ittmm-2023.pdf. (Дата обращения: 10 ноября 2024).
3. *Борисов Д.В., Веселовский А.В.* Кинетика связывания лиганда с рецептором в разработке лекарств // Биомедицинская химия. 2020. **66**, № 1. 42–53. doi 10.18097/PBMC20206601042.
4. *Souza P.C.T., Thallmair S., Confitti P., et al.* Protein–ligand binding with the coarse-grained Martini model // Nature Communications. 2020. **11**, Article Number 3714 (2020). doi 10.1038/s41467-020-17437-5.
5. *Qin S., Pang X., Zhou H.-X.* Automated prediction of protein association rate constants // Structure. 2011. **19**, N 12. 1744–1751. doi 10.1016/j.str.2011.10.015.
6. *Alsallaq R., Zhou H.-X.* Electrostatic rate enhancement and transient complex of protein–protein association // Proteins. 2008. **71**, N 1. 320–335. doi 10.1002/prot.21679.
7. *Dhusia K., Su Z., Wu Y.* Using coarse-grained simulations to characterize the mechanisms of protein–protein association // Biomolecules. 2020. **10**, N 7. Article Number 1056. doi 10.3390/biom10071056.

8. *Rogal J., Bolhuis P.G.* Multiple state transition path sampling // J. Chem. Phys. 2008. **129**, N 22. Article Number 224107. doi 10.1063/1.3029696.
9. *Jankauskaité J., Jiménez-Garcia B., Dapkünas J., et al.* SKEMPI 2.0: an updated benchmark of changes in protein–protein binding energy, kinetics and thermodynamics upon mutation // Bioinformatics. 2019. **35**, N 3. 462–469. doi 10.1093/bioinformatics/bty635.
10. *Berman H.M., Westbrook J., Feng Z., et al.* The protein data bank // Nucleic Acids Res. 2000. **28**, N 1. 235–242. doi 10.1093/nar/28.1.235.
11. *Сысоев А.В., Гориков А.В., Волокитин В.Д., и др.* Учебный курс "Программирование с использованием модели oneAPI" // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. **11**, № 3. 45–58. doi 10.14529/cmse220303.
12. *Adam Gh., Bashashin M., Belyakov D., et al.* IT-ecosystem of the HybriLIT heterogeneous platform for high-performance computing and training of IT-specialists // Proc. 8th Int. Conf. on Distributed Computing and Grid-Technologies in Science and Education, Dubna, Russia, September 10–14, 2018. CEUR Workshop Proc. 2018. **2267**. 638–644. https://ceur-ws.org/Vol-2267/638-644-paper-122.pdf. Cited November 10, 2024.
13. *E W., Ren W., Vanden-Eijnden E.* Simplified and improved string method for computing the minimum energy paths in barrier-crossing events // J. Chem. Phys. 2007. **126**, N 16. Article Number 164103. doi 10.1063/1.2720838.
14. *Sahni S., Vairaktarakis G.* The master–slave paradigm in parallel computer and industrial settings // Journal of Global Optimization. 1996. **9**, N 3–4. 357–377. doi 10.1007/BF00121679.
15. Программное средство Valgrind. https://valgrind.org. (Дата обращения: 10 ноября 2024).
16. Репозиторий с синтетическим тестом и фрагментами кода. https://vcs.uni-dubna.ru/psm/multithreading. (Дата обращения: 10 ноября 2024).
17. *Williams S., Waterman A., Patterson D.* Roofline: an insightful visual performance model for multicore architectures // Communications of the ACM. 2009. **52**, N 4. 65–76. doi 10.1145/1498765.1498785.
18. *Voss M., Asenjo R., Reinders J.* Pro TBB: C++ parallel programming with threading building blocks. Berkeley: Apress, 2019. doi 10.1007/978-1-4842-4398-5.

Поступила в редакцию
9 октября 2024 г.

Принята к публикации
5 ноября 2024 г.

Информация об авторах

Сергей Владимирович Полуян — старший преподаватель; Государственный университет "Дубна", Институт системного анализа и управления, ул. Университетская, 19, 141982, Дубна, Российская Федерация.

Николай Михайлович Ершов — к.ф.-м.н., старший научный сотрудник; Московский государственный университет имени М. В. Ломоносова, факультет вычислительной математики и кибернетики, Ленинские горы, 1–52, 119991, Москва, Российская Федерация.

References

1. S. S. Khruschev, A. M. Abaturova, A. N. Diakonova, et al., "Multi-Particle Brownian Dynamics Software ProKSim for Protein–Protein Interactions Modeling," Comput. Res. Model. **5** (1), 47–64 (2013). doi 10.20537/2076-7633-2013-5-1-47-64.
2. S. V. Poluyan, D. A. Nikulin, and N. M. Ershov, "Development and Verification of a Score Function for Estimation of Intermolecular Interactions in Protein–Protein Complexes," in *Proc. Int. Conf. on ITTMM, Moscow, Russia, April 17–21, 2023* (RUDN Univ., Moscow, 2023), pp. 231–235. https://events.rudn.ru/event/198/attachments/550/1474/ittmm-2023.pdf. Cited November 10, 2024.
3. D. V. Borisov and A. V. Veselovsky, "Ligand–Receptor Binding Kinetics in Drug Design," Biomeditsinskaya Khimiya **66** (1), 42–53 (2020). doi 10.18097/PBMC20206601042.
4. P. C. T. Souza, S. Thallmair, P. Confitti, et al., "Protein–Ligand Binding with the Coarse-Grained Martini Model," Nat. Commun. **11**, Article Number 3714 (2020). doi 10.1038/s41467-020-17437-5.
5. S. Qin, X. Pang, and H.-X. Zhou, "Automated Prediction of Protein Association Rate Constants," Structure **19** (12), 1744–1751 (2011). doi 10.1016/j.str.2011.10.015.

2024, **25** (4), 453–463. doi 10.26089/NumMet.v25r434

6. R. Alsallaq and H.-X. Zhou, "Electrostatic Rate Enhancement and Transient Complex of Protein–Protein Association," Proteins **71** (1), 320–335 (2008). doi 10.1002/prot.21679.
7. K. Dhusia, Z. Su, and Y. Wu, "Using Coarse-Grained Simulations to Characterize the Mechanisms of Protein–Protein Association," Biomolecules **10** (7), Article Number 1056 (2020). doi 10.3390/biom10071056.
8. J. Rogal and P. G. Bolhuis, "Multiple State Transition Path Sampling," J. Chem. Phys. **129** (22), Article Number 224107 (2008). doi 10.1063/1.3029696.
9. J. Jankauskaité, B. Jiménez-Garcia, J. Dapkünas, et al., "SKEMPI 2.0: An Updated Benchmark of Changes in Protein–Protein Binding Energy, Kinetics and Thermodynamics upon Mutation," Bioinformatics **35** (3), 462–469 (2019). doi 10.1093/bioinformatics/bty635.
10. H. M. Berman, J. Westbrook, Z. Feng, et al., "The Protein Data Bank," Nucleic Acids Res. **28** (1), 235–242 (2000). doi 10.1093/nar/28.1.235.
11. A. V. Sysoyev, A. V. Gorshkov, V. D. Volokitin, et al., "Programming with oneAPI: New Course Heterogeneous Computing," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **11** (3), 45–58 (2022). doi 10.14529/cmse220303.
12. Gh. Adam, M. Bashashin, D. Belyakov, et al., "IT-Ecosystem of the HybriLIT Heterogeneous Platform for High-Performance Computing and Training of IT-Specialists," in Proc. 8th Int. Conf. on Distributed Computing and Grid-Technologies in Science and Education, Dubna, Russia, September 10–14, 2018. CEUR Workshop Proc. **2267**, 638–644 (2018). https://ceur-ws.org/Vol-2267/638-644-paper-122.pdf. Cited November 10, 2024.
13. W. E, W. Ren, and E. Vanden-Eijnden, "Simplified and Improved String Method for Computing the Minimum Energy Paths in Barrier-Crossing Events," J. Chem. Phys. **126** (16), Article Number 164103 (2007). doi 10.1063/1.2720838.
14. S. Sahni and G. Vairaktarakis, "The Master–Slave Paradigm in Parallel Computer and Industrial Settings," J. Glob. Optim. **9** (3–4), 357–377 (1996). doi 10.1007/BF00121679.
15. Programming Tool Valgrind. https://valgrind.org. Cited November 10, 2024.
16. Repository with Synthetic Test and Code Fragments. https://vcs.uni-dubna.ru/psm/multithreading. Cited November 10, 2024.
17. S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," Commun. ACM **52** (4), 65–76 (2009). doi 10.1145/1498765.1498785.
18. M. Voss, R. Asenjo, and J. Reinders, *Pro TBB: C++ Parallel Programming with Threading Building Blocks* (Apress, Berkeley, 2019). doi 10.1007/978-1-4842-4398-5.

Received
October 9, 2024

Accepted for publication
November 5, 2024

Information about the authors

Sergey V. Poluyan — Senior Teacher; Dubna State University, Institute of System Analysis and Management, Universitetskaya ulitsa 19, 141982, Dubna, Russia.

Nikolay M. Ershov — Ph.D., Senior Research Fellow; Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics, Leninskie Gory, 1–52, 119991, Moscow, Russia.