Использование параллельных вычислений для эффективного построения множественных выравниваний структур белков

Вл.В. Воеводин[1], М.В. Шегай[1], Н.Н. Попова[1], Д.А. Суплатов[1], В.К. Швядас[1]
Московский государственный университет имени М.В.Ломоносова[1]

Множественное пространственное выравнивание структур белков является важным инструментом структурной биологии. Анализ структур белков позволяет устанавливать их гомологию, т.е происхождение от общего предка. Бурный рост количества известных белковых структур обуславливает требования к скорости работы алгоритмов пространственного выравнивания.
В данной работе предлагается стратегия использования параллельных вычислений для эффективного построения множественного пространственного выравнивания с использовпнием суперкомпьютеров. Разработанный подход основан на хорошо зарекомендовавшем себя последовательном методе пространственного выравнивания белковых структур Matt (Multiple Alingment with Translations and Twists). Реализация Matt с использованием параллельных вычислений позволяет ускорить построение выравниваний без потери качества. В работе приводятся оценки эффективности использования параллельных вычислений и анализируются результаты вычислительных экспериментов, проведенных на суперкомпьютере "Ломоносов".

Ключевые слова: биоинформатика, множественное выравнивание, пространственное выравнивание, Matt.

1. Введение

В основе структуры любого организма и всех протекающих в нем жизненных реакций лежат белки. Любые изменения в этих белках приводят к изменению жизненных процессов, протекающих в организме. Белки являются нерегулярными полимерами, т.е. молекулами, функции которых существенно определяются числом, составом и порядком расположения входящих в них мономеров. В начале 50-ых годов прошлого века было показано, что белковая цепь имеет уникальную последовательность звеньев – аминокислотных остатков ("остаток" – это то, что осталось от свободной аминокислоты после ее встраивания в белковую цепь). Эта цепь имеет химически регулярный остов ("главную цепь"), от которого отходят разнообразные боковые группы аминокислот. Белки укладываются в сложные несимметричные трехмерные (третичные) структуры.

Пространственное выравнивание позволяет изучать взаимосвязь между структурой и функциями белков. Пространственное выравнивание – метод установления сходства организации, а также эволюционных взаимосвязей между двумя или более белковыми структурами, основанный на сравнении их третичных структур. Целью пространственного выравнивания является наилучшее геометричесчкое сопоставление элементов третичной структуры.

На рисунке 1 представлен пример выравнивания пары белков тиоредоксинов человека(номер структуры в PDB: 3TRX) и дрозофилы (1XWC). Из рисунка видно, что значительные участки структур этих белков, которые соответствуют фрагменту предковой укладки (т.н. трехслойному сэндвичу), после применения геометрических преобразований (трансляций и вращений цельных структур) совмещаются с высокой степенью сходства. Это позволяет говорить об общем происхождении этих белков в эволюции, а также дает возможность сравнивать механизмы их действия.

Системный биоинформатический анализ выравниваний семейств родственных белков

позволяет найти структурные изменения, необходимые для конструирования новых белков с заданными свойствами [1]. Кроме того, сходство третичных структур можно использовать для предсказания функций малоизученных белков. Благодаря структурному выравниванию можно определить эволюционно эквивалентные аминокислотные остатки, если выравниваемые белки имеют общего предка. Помимо этого, если у схожих белков наблюдаются консервативные области, в которых структура практически не изменяется, то это может указывать на функциональную либо структурную значимость этой области.

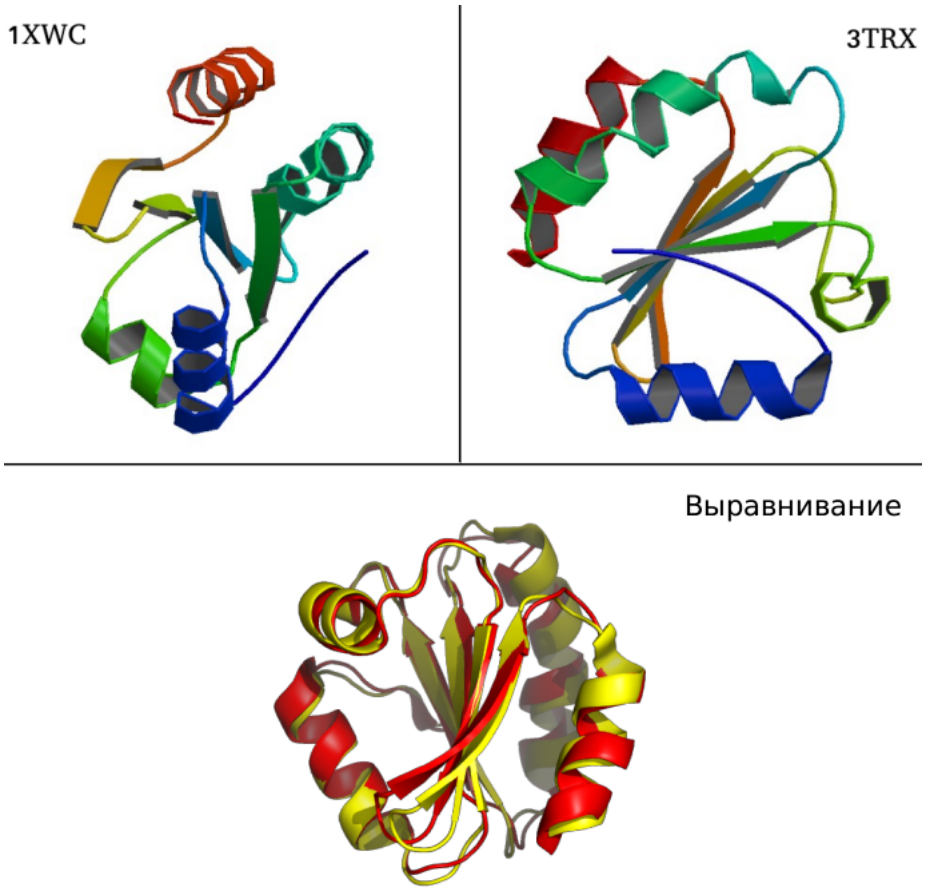

Рис. 1. Выравнивание белков тиоредоксинов дрозофилы Drosophila melanogaster (1XWC) и человека(3TRX). Красной лентой в выравнивании представлен белок человека, желтой – дрозофилы

Программы множественного структурного выравнивания обычно строятся на основе методов попарного пространственного выравнивания. Даже упрощенные версии алгоритмов пространственного выравнивания являются NP-сложными[2, 3].

Методы попарного пространственного выравнивания можно разделить на 3 класса:

1. Методы, основанные на выравненных парах фрагментов структур (*AFP, aligned fragment pair*)[4, 5]. Эти методы используют короткие фрагменты из обеих белковых структур, производят над ними некоторые преобразования и собирают их в геометрически допустимую структуру.

2. Методы, рассматривающие попарные расстояния отдельно внутри каждой структуры разыскивающие наибольшее множество аминокислотных остатков, имеющих похожие попарные расстояния в обеих структурах.

3. Все остальные методы, не входящие в первые два пункта.

Классический геометрический подход измерения качества структурного выравнивания белков включает в себя два параметра: число аминокислотных остатков, которые используются в выравнивании и среднеквадратичное отклонение расстояний между атомами (*RMSD*), находящимися в сохраняемом ядре[1]

Следует учитывать, что на сегодняшний день не существует универсальной метрики оценки качества выравнивания и зачастую только экспертная оценка является решающей в выборе того или иного метода выравнивания. Matt является хорошо зарекомендовавшим себя методом множественного выравнивания.

Качество выравнивания не является единственным критерием выбора программы множественного выравнивания. Бурный рост количества известных белковых структур обуславливает требования к скорости работы программ множественного выравнивния и количеству обрабатываемых ими последовательностей.

2. Множественное выравнивание с поворотами и сдвигами (Matt)

Matt (multiple alignment with translation and twists, множественное выравнивание с поворотами и сдвигами) – алгоритм, основанный на методе выравненных пар участков структур (AFP)[7]. Отличительной особенностью Matt является ослабление жесткости главной цепи белка. Это ослабление позволяет структуре белка изгибаться и вращаться для получения нужного выравнивания.

Matt показывает лучшие результаты на стандартных проверочных наборах данных (HOMSTRAD[8], SABmark[9]) по сравнению с известными программами множественного выравнивания.

В качестве входных данных Matt использует файлы в формате PDB[10], описывающие белки, подлежащие выравниванию. Результат множественного выравнивания на выходе Matt также сохраняется в файл формата PDB.

Блок-схема алгоритма представлена на рисунке 2. На вход программе Matt подается множество из *g* структур. Изначально каждая структура образует отдельную группу.

Итеративная часть программы состоит из *g-1* итераций. На каждой итерации две группы выравненных структур сливаются в одну, образуя новое выравнивание. В итеративной части Matt допускаются геометрически недопустимые изменения в структуре. После завершения итеративной части производится исправление глобального выравнивания, в результате которого получается геометрически допустимая структура.

В итеративной части последовательной программы Matt можно выделить три основные стадии.

2.1. Пары фрагментов

Действия первой стадии аналогичны тому, что выполняется многими методами, основанными на AFP. Matt рассматривает фрагменты от 5 до 9 смежных аминокислотных остатков. Парой фрагментов будем считать 2 фрагмента одинаковой длины. Оценка выравнивания пары фрагментов, по одному из каждой группы, высчитывается на основе преобразований, производимыми над всеми структурами группы. Для каждой пары фрагментов высчитывается оценка выравнивания.

2.2. Сборка с поворотами и сдвигами

Основное нововведение Matt заключается в способе сборки коротких фрагментов в глобальное выравнивание. Matt использует динамическое программирование для получения на каждой итерации более длинных групп выравненных фрагментов. Оценка качества выров-

[1]Сохраняемое ядро (*conserved core*)[6] – множество аминокислотных остатков, которые консервативны (т.е. совпадают) по расположению основной цепи во всех рассматриваемых белках.

нивания основывается на сумме оценок выравниваний каждого выравненного фрагмента и штрафах за геометрические преобразования[1] главной цепи одного белка в другую.

Matt находит пару групп с лучшей оценкой качества и собирает их в новое множественное выравнивание, объединяя две группы. Если остается ровно одна группа, то алгоритм переходит в финальную стадию, иначе выполняется стадия корректировки и расширения (realign and extend phase) и выполняется переход на следующую итерацию.

2.3. Стадия корректировки и расширения

Стадия корректировки не изменяет взаимное расположение аминокислотных остатков в множественном выравнивании. На этой стадии алгоритм пытается найти локальные преобразования, которые бы уменьшили RMSD выравненных фрагментов в только что объединенной группе.

В стадии расширения множественное выравнивание расширяется в направлении обоих концов каждого из фрагмнетов настолько, насколько позволяет порог RMSD. На этой стадии допустимо наложение до 5 аминокислотных фрагментов в расширенных фрагментах.

2.4. Финальный проход

В финальном проходе производится исправление глобального выравнивания, которое оптимизирует RMSD и строит геометрически допустимое выравнивание. Для этого используется метод, предложенный Бартоном и Штернбергом (Barton, Sternberg)[11].

Сложность последовательного алгоритма Matt оценивается как $O(k^2n^3log(n))$, где k – число последовательностей, используемых в выравнивании, а n – длина самой длинной последовательности.

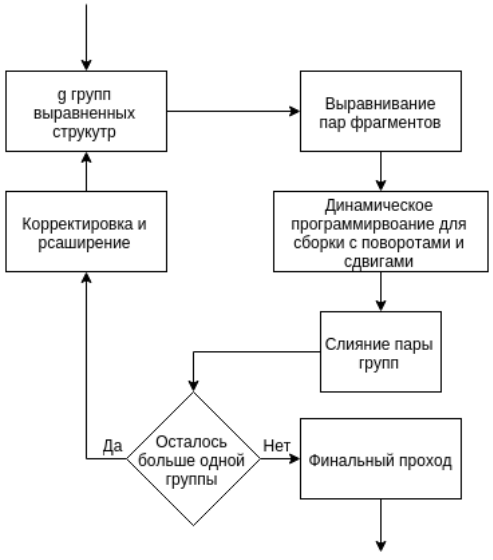

Рис. 2. Схема алгоритма Matt

.

3. Параллельный алгоритм множественного выравнивания

На основе последовательного алгоритма Matt предложена и реализована стратегия использования параллельных вычислений для эффективного построения множественных структурных выравниваний белков с использованием суперкомпьютера.

[1]Величины штрафов вычислены эмпирически на основе базы данных Homstrad.

3.1. Возможность распараллеливания

```
Function slave_routine()
    /* Первый этап                                                    */
    scatter(alignment_pairs, als, to_slaves, from_master)
    for i = 0 to als.length do
        aligned_alignments[i] = align_alignments(als[i])
    end
    gather(als, aligned_alignments, from_slaves, to_master)

    /* Второй этап                                                    */
    while true do
        recieve(terminator, from_master)
        if terminator then
            break
        end

        broadcast(ma, from_master)
        scatter(alignment_pairs, als, to_slaves, from_master)
        for i = 0 to als.length do
            aligned_alignments[i] = align_alignments(als[i])
        end
    end
    gather(als, aligned_alignments, from_slaves, to_master)
```

Рис. 3. Псевдокод первого и второго этапов параллельной реализации алгоритма Matt для рабочих процессов

В программе Matt можно выделить четыре основных этапа: ввод данных, предварительная обработка всевозможных пар, итеративная часть, финальный проход. При этом большая часть вычислений производится на этапе предварительной обработки всевозможных пар, когда рассматриваются всевозможные выравнивания пар начальных структур и в итеративной части алгоритма, когда алгоритм выбирает лучшие пары для выравнивания.

Каждую пару структур, формируемую на первом этапе алгоритма, можно получить независимо от остальных, что создает хорошую возможность для распараллеливания этого участка программы. В итеративной же части алгоритма на каждой итерации отдельно друг от друга определяются выравнивания для n структур (n – параметр, зависящий от номера итерации).

В Matt за обработку всевозможных пар и итеративную часть алгоритма отвечает функция Align(). Профилировка оригинальной версии программы показала, что в функции Align() на этапе предварительной обработки пар и в итеративной части для выравнивания используется одна и та же функция AlignAlignments(), вычисления которой и занимают основное время работы программы (см. рис.3).

3.2. Структура и схема распараллеливания

Для распараллеливания вычислений на основе алгоритма Matt используется метод коллективного решения.

Мастер-процесс на первом этапе алгоритма получает всевозможные пары структур, сериализует[1] и равномерно распределяет их между всеми доступными процессами. Далее

[1]Сериализация – перевод структуры данных языка в непрерывный участок байтов.

```
Function master_routine(alignments)
    /* Первый этап                                                    */
    alignment_pairs = generate(alignments)        /* n(n−1)/2 всевозможных пар */
    scatter(alignment_pairs, als, to_slaves, from_master)
    for i = 0 to als.length do
        aligned_alignments[i] = align_alignments(als[i])
    end
    gather(als, aligned_alignments, from_slaves, to_master)

    /* Второй этап                                                    */
    while aligned_alignments.length > 0 do
        ma = find_best_alignment(aligned_alignments)

        /* Удалить все, что содержит ma из обоих множеств             */
        remove (ma, alignments)
        remove (ma, aligned_alignments)

        number_of_slaves = min(number_of_slaves, alignments.length)
        for slave_id = 0 number_of_slaves do
            send(terminator, slave_id)
        end

        broadcast(ma, from_master)
        scatter(alignment_pairs, als, to_slaves, from_master)

        naa = aligned_alignments.length
        for i = 0 to als.length do
            aligned_alignments[naa + i] = align_alignments(ma, als[i])
        end
        gather(als, aligned_alignments + naa, from_slaves, to_master)
    end
    return alignments[0]
```

Рис. 4. Псевдокод первого и второго этапов параллельной реализации алгоритма Matt для мастер-процесса

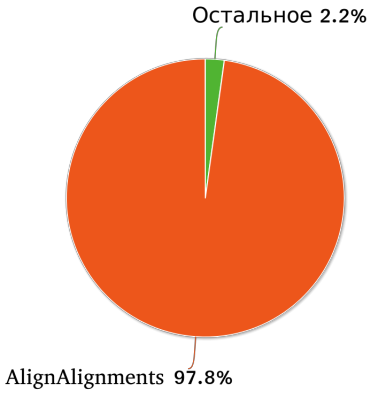

Рис. 5. Структура временных затрат Matt

происходит выполнение функции AlignAlignments всеми процессами(в том числе, и мастер-

процессом), сбор данных и десериализация[1] на мастер-процессе.

На каждой итерации второго этапа происходит выбор выравнивания с лучшей оценкой среди уже выравненных структур, сериализация и равномерное распределение n структур и лучшего выравнивания между всеми доступными процессами. Далее происходит выполнение функции AlignAlignments всеми процессами (в том числе и мастер-процессом), сбор данных и их десериализация на мастер-процессе.

В статье представлен псевдокод первого и второго этапов параллельного алгоритма (см. рис. 3, 4).

3.3. Используемые технологии и инструменты

Распараллеливание вычислений программы Matt выполнено с использованием технологии MPI[12]. *Message Passing Interface* (MPI, интерфейс передачи сообщений) – программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между параллельными процессами, выполняющими одну задачу.

Для анализ временных затрат последовательного кода Matt использовался профилировщик gperftools[13].

Сравнение результатов работы последовательного варианта и параллельной реализации программы Matt проводилась с использованием стандартной Unix-утилиты diff.

4. Оценка эффективности

Исследование разработанного алгоритма проводится согласно классическим определениям ускорения и эффективности.

Ускорением S_p параллельного алгоритма называют отношение времени T_1 выполнения последовательной программы ко времени T_p выполнения параллельной программы на p процессорах

$$S_p = \frac{T_1}{T_p} \tag{1}$$

Пусть n – число цепочек, время выравнивания одной пары постоянно и равно t_0, τ_s – время работы программы в нераспараллеливаемой части, накладные расходы на пересылку одной пары постоянны и равны τ_m.

Тогда время работы последовательной версии Matt будет равно

$$T_1 = \tau_s + \frac{n(n-1)}{2}t_0 + \sum_{k=1}^{n-1} kt_0 = \tau_s + t_0(\frac{n(n-1)}{2} + \frac{n(n-1)}{2})$$

Таким образом,

$$T_1 = \tau_s + t_0 n(n-1) \tag{2}$$

Время работы параллельного алгоритма на p процессорах

$$\begin{gathered} T_p = \tau_s + \frac{n(n-1)}{2}(\frac{t_0}{p} + \tau_m) + \sum_{k=p}^{n-1} k(\frac{t_0}{p} + \tau_m) + \sum_{k=1}^{p-1} k\tau_m + (p-1)t_0 = \\ = \tau_s + \frac{t_0}{p}n(n-1) + \frac{t_0}{2}(p-1) + \tau_m n(n-1) \end{gathered} \tag{3}$$

[1]Десериализация – обратный процесс к сериализации, т.е восстановление структуры данных языка из непрерывной последовательности байтов.

Подставим (2) и (3) в (1)

$$S_p = \frac{T_1}{T_p} = \frac{\tau_s + t_0 n(n-1)}{\tau_s + \frac{t_0}{p} n(n-1) + \frac{t_0}{2}(p-1) + \tau_m n(n-1)} \tag{4}$$

Эффективностью параллельного алгоритма называется величина

$$E_p = \frac{S_p}{p} \tag{5}$$

В (4) положим, что $\tau_s = 0$ и положим, что $\tau_m = \alpha t_0 (\alpha > 0)$[1], тогда получим

$$\begin{gathered} S_p = \frac{t_0 n(n-1)}{\frac{t_0}{p} n(n-1) + \frac{t_0}{2}(p-1) + \alpha t_0 n(n-1)} = \\ = \frac{2pn(n-1)}{2n(n-1) + p(p-1) + 2p\alpha n(n-1)} = \frac{2pn(n-1)}{2n(n-1)(1+\alpha p) + p(p-1)} \end{gathered} \tag{6}$$

Тогда

$$E_p = \frac{2n(n-1)}{2n(n-1)(1+\alpha p) + p(p-1)} = \frac{1}{1 + \alpha p + \frac{p(p-1)}{2n(n-1)}} \tag{7}$$

Формулу (7) можно использовать для оценки масштабируемости параллельного алгоритма, а также для подбора требуемого количества вычислительных узлов для решения задачи с заданной эффективностью при известном объемом входных данных.

5. Вычислительные эксперименты

Исследование разработанного параллельного алгоритма проводилось на суперкомпьютере "Ломоносов", входящем в состав суперкомпьютерного комплекса МГУ имени М.В. Ломоносова[14]. Вычислительный эксперимент проводился на вычислительных узлах, построенных на базе процессоров Intel® Xeon 5570 с 12Гб оперативной памяти.

Для проверки корректности работы программы сравнивались PDB файлы, полученные на выходе оригинальной последовательной программы и параллельной программы.

В качестве входных данных брались белковые структуры, каждая из которых состояла примерно из 3000 атомов. Всего было использовано 346 структур, из них было составлено 3 множества структур, по 192 стрктуры в каждом. На каждом из трех множеств программа запускалась 5 раз для заданного числа процессоров. Результаты 15 запусков усреднялись.

Как отмечалось ранее, коэффициент α в формуле (7) зависит от длины последовательности и параметров конкретной вычислительной системы. Для суперкомпьютера "Ломоносов" найдена $\alpha = 0.0003$ при длине последовательности около 3000.

Как видно из графика (см. рис. 6), характеры поведения теоретической и экспериментальной кривых совпадают, что подтверждает справедливость предложенной оценки эффективности.

6. Заключение

В статье предложен и реализован параллельный алгоритм множественного выравнивания белков, основанный на программе Matt. Получаемые белковые структруры на выходе данной реализации в точности совпадают со структурами, полученными оригинальным последовательным алгоритмом. При этом параллельная реализация позволяет существенно снизить время, затрачиваемое на нахождение множественного выравнивания.

[1]Коэффициент α зависит от длины последовательности и параметров конкретной вычислительной системы (скорости передачи данных и вычислений).

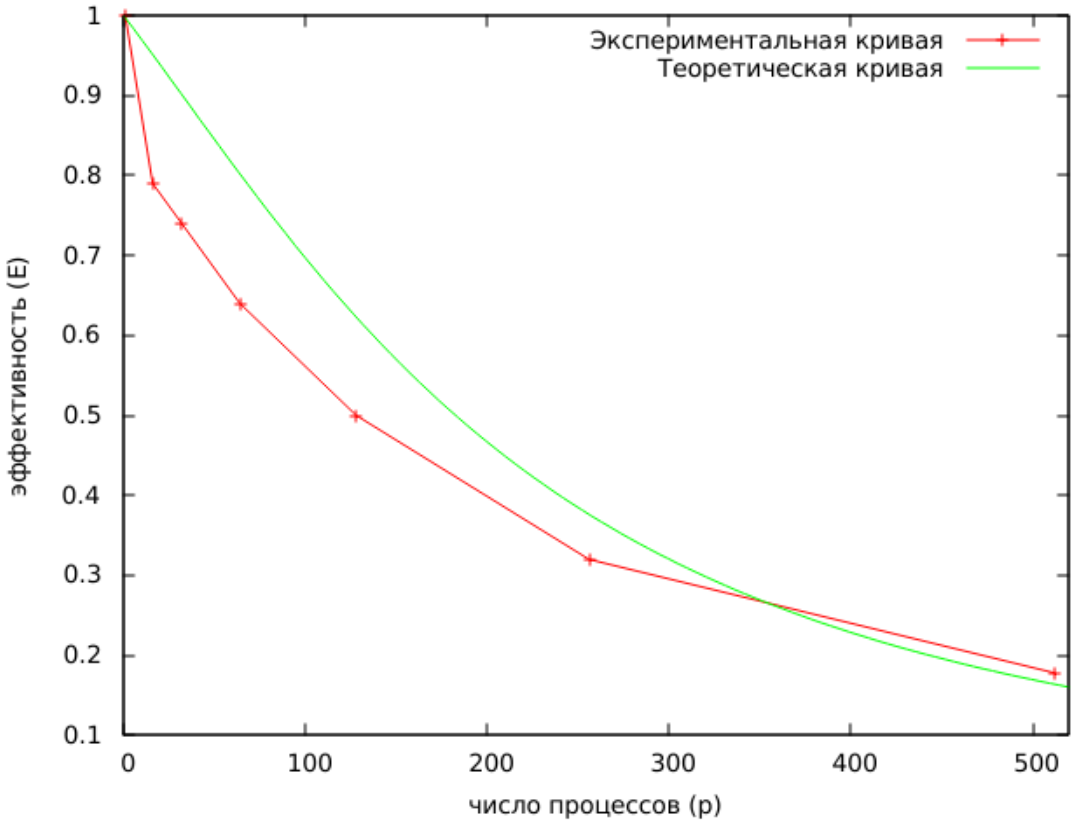

Рис. 6. Эффективность parMatt

Таблица 1. Результаты запусков параллельной программы на суперкопмьютере "Ломоносов"

p	Время[c]	S_p	Экспериментальная E_p^i	Теоретическая E_p^t	$\lvert E_p^i - E_p^t \rvert$
1	111687	1	1		
16	8731	12.79	0.79	0.95	0.26
32	4677	23.88	0.74	0.90	0.16
64	2727	40.95	0.64	0.80	0.16
128	1730	64.55	0.50	0.62	0.12
256	1341	83.28	0.32	0.37	0.05
512	1220	91.54	0.17	0.16	0.01

Приведена теоретическая оценка эффективности разработанного параллельного алгоритма. Предложенная оценка подтверждена экспериментальными результатами. Оценка эффективности позволяет определить оптимальное количество вычислительных узлов для решения задачи с заданным объемом входных данных.

Дальнейшее усовершенствование параллельного алгоритма будет направлено на снятие ограничений существующей реализации на количество обрабатываемых структур. В настоящее время количество обрабатываемых структур определяется объемом оперативной памяти вычислительного узла, на котором выполняется мастер-процесс. Проведенный экспе-

римент с использованием вычислительных узлов суперкомпьютера "Ломоносов" с большой оперативной памятью показывает возможность обработки значительного числа (тысячи) структур за приемлемое время.

Работа выполнена при поддержке гранта РФФИ № 14-07-0437.

Литература

1. Suplatov D.A., Voevodin V.V., Švedas V.K. Robust enzyme design: Bioinformatic tools for improved protein stability // Biotechnology journal, Wiley - VCH Verlag GmbH & CO.KGaA (Germany), Vol. 10, No 3, P. 344–355 DOI
2. Goldman D, Istrail S, Papadimitriou CH (1999) Algorithmic aspects of protein structure similarity. Proceedings of the 40th Annual Symposium on Foundations of Computer Science. Los Alamitos (California): IEEE Computer Society. P. 512–522.
3. Wang L, Jiang T (1994) On the complexity of multiple sequence alignment // Journal of computational biology 1, No. 4 (1994), P. 337–348.
4. Shindyalov I, Bourne P (1998) Protein structure alignment by incremental combinatorial extension (CE) of the optimal path // Protein engineering 11, No. 9 (1998), P. 739–747.
5. Ye Y, Godzik A (2003) Flexible structure alignment by chaining aligned fragment pairs allowing twists // Bioinformatics 19, No. suppl 2 (2003), P. ii246–ii255.
6. Eidhammer I, Jonassen I, Taylor WR (2000) Structure comparison and structure patterns // Journal of Computational Biology 7, No. 5 (2000), P. 685–716.
7. M. Menke, B. Berger, L. Cowen, "Matt: Local Flexibility Aids Protein Multiple Structure Alignment"// PLoS Comput Biol 4, No. 1 (2008), P. e10.
8. Mizuguchi K, Deane C, Blundell TL, Overington J (1998) HOMSTRAD: A database of protein structure alignments for homologous families // Protein science: a publication of the Protein Society 7, No. 11 (1998), P. 2469–2471.
9. VanWalle I, Lasters I, Wyns L (2005) SABmark—A benchmark for sequence alignment that covers the entire known fold space // Bioinformatics 21, No. 7 (2005), P. 1267–1268.
10. PDB File Format. URL: `http://www.rcsb.org/pdb/static.do?p=file_formats/pdb/index.html` (дата обращения: 25.12.2015)
11. Barton G, Sternberg M (1987) A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons // Journal of molecular biology 198, No. 2 (1987), P. 327–337.
12. Message Passing Interface (MPI) Forum Home Page. URL: `http://www.mpi-forum.org/` (дата обращения: 25.12.2015)
13. gperftools. URL: `https://github.com/gperftools/gperftools` (дата обращения: 25.12.2015)
14. Воеводин Вл.В., Жуматий С.А., Соболев С.И., Антонов А.С., Брызгалов П.А., Никитенко Д.А., Стефанов К.С., Воеводин Вад.В. Практика суперкомпьютера "Ломоносов". // Открытые системы. - Москва: Издательский дом "Открытые системы № 7, 2012. С. 36–39.

Use of parallel computing in effective protein multiple structure alignment

Vl.V. Voevodin[1], M.V. Shegay[1], N.N. Popova[1], D.A. Suplatov[1], V.K. Švedas[1]
Lomonosov Moscow State University[1]

Protein multiple structure alignment is a valuable tool of structural biology. Protein structure analysis can be used to detect homology between proteins; i.e, the existence of shared ancestry. Large number of known protein structures sets up the requirements for the algorithms of structural alignment.
A strategy of parallel computations for effective construction of multiple structure alignment with use of supercomputers is introduced in this paper. The suggested approach is based on Matt (Multiple Alignment with Translations and Twists), a well-known method of protein multiple alignment.
The parallelized Matt implementation significantly speeds up construction of alignment without quality loss. This paper provides estimates for the efficiency of parallel computations and analisys of computational experiments carried out on Lomonosov supercomputer.

Keywords: bioinformatics, multiple alignment, structural alignment, Matt.

References

1. Suplatov D.A., Voevodin V.V., Švedas V.K. Robust enzyme design: Bioinformatic tools for improved protein stability // Biotechnology journal, Wiley - VCH Verlag GmbH & CO.KGaA (Germany), Vol. 10, No 3, P. 344–355 DOI

2. Goldman D, Istrail S, Papadimitriou CH (1999) Algorithmic aspects of protein structure similarity. Proceedings of the 40th Annual Symposium on Foundations of Computer Science. Los Alamitos (California): IEEE Computer Society. P. 512–522.

3. Wang L, Jiang T (1994) On the complexity of multiple sequence alignment // Journal of computational biology 1, No. 4 (1994), P. 337–348.

4. Shindyalov I, Bourne P (1998) Protein structure alignment by incremental combinatorial extension (CE) of the optimal path // Protein engineering 11, No. 9 (1998), P. 739–747.

5. Ye Y, Godzik A (2003) Flexible structure alignment by chaining aligned fragment pairs allowing twists // Bioinformatics 19, No. suppl 2 (2003), P. ii246–ii255.

6. Eidhammer I, Jonassen I, Taylor WR (2000) Structure comparison and structure patterns // Journal of Computational Biology 7, No. 5 (2000), P. 685–716.

7. M. Menke, B. Berger, L. Cowen, "Matt: Local Flexibility Aids Protein Multiple Structure Alignment"// PLoS Comput Biol 4, No. 1 (2008), P. e10.

8. Mizuguchi K, Deane C, Blundell TL, Overington J (1998) HOMSTRAD: A database of protein structure alignments for homologous families // Protein science: a publication of the Protein Society 7, No. 11 (1998), P. 2469–2471.

9. VanWalle I, Lasters I, Wyns L (2005) SABmark—A benchmark for sequence alignment that covers the entire known fold space // Bioinformatics 21, No. 7 (2005), P. 1267–1268.

10. PDB File Format. URL: `http://www.rcsb.org/pdb/static.do?p=file_formats/pdb/index.html` (accessed: 25.12.2015)

11. Barton G, Sternberg M (1987) A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons // Journal of molecular biology 198, No. 2 (1987): 327-337.

12. Message Passing Interface (MPI) Forum Home Page. URL: `http://www.mpi-forum.org/` (accessed: 25.12.2015)

13. gperftools. URL: `https://github.com/gperftools/gperftools` (accessed: 25.12.2015)

14. Voevodin Vl.V., Zhumatiy S.A., Sobolev S.I., Antonov A.S., Bryzgalov P.A., Nikitenko D.A., Stefanov K.S., Voevodin Vad.V. Praktika superkomp'yutera "Lomonosov"[Practices of Lomonosv Supercomputer] // Otkrytye sistemy. - Moskva: Izdatel'skiy dom "Otkrytye sistemy"[Open systems. - Moscow: Open Systems Publications], No. 7, 2012. P. 36–39.