CLUSTALW2: ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ С ПОМОЩЬЮ ТЕХНОЛОГИИ OPENMP

И.С. Пироженко, А.Н. Сальников

Введение

Алгоритмы множественного выравнивания представляют собой инструмент для установления функциональных, структурных или эволюционных взаимосвязей между биологическими последовательностями. Несмотря на то, что задача множественного выравнивания была сформулирована более 20 лет назад[1], она не утрачивает своей актуальности, ведь набор алгоритмов множественного выравнивания это важный инструмент, используемый биологами повсеместно. Сложность задачи множественного выравнивания обусловливается экспоненциальным ростом времени счета не параллельной программы при увеличении либо числа биологических последовательностей, либо их длины[1]. Даже использование эвристик в алгоритме множественного выравнивания дает NP-полную задачу. Поэтому актуальна задача применения высокопроизводительных вычислений, и параллельная реализация существующих алгоритмов для ускорения времени счета программ множественного выравнивания.

Существует довольно много широко распространенных последовательных алгоритмов множественного выравнивания, таких как: ClustalW2 (Gibson T., Thompson J., Higgins D.), T-Coffee (Cédric Notredame), MUSCLE (Edgar R.C.), MAFFT (Kazutaka Katoh). Из всех перечисленных алгоритмов, ClustalW2 обладает рядом преимуществ. Т.к. он показывает хорошие результаты на тестах качества выравнивания [2], имеет понимаемый код, доступный всем желающим по лицензии LGPL.

Активное внедрение многопроцессорных и многоядерных архитектур дает возможность значительно сократить время работы молекулярного биолога над своей проблемой. Задача ускорения алгоритма ClustalW2 с помощью «распараллеливания» рассмотрена в этой работе.

Множественное выравнивание

Определим две последовательности символов следующим образом. Последовательности $S_1 = s_{1,1} s_{1,2} \dots s_{1,L_1}$ и $S_2 = s_{2,1} s_{2,2} \dots s_{2,L_2}$ алфавита $A = \{a_1 \dots a_n\}$. Матрица $R = ((ri, j))$ размерности $2 \times L$ называется парным выравниванием последовательностей S_1 и S_2 , если каждая позиция ri, j матрицы содержит либо символ алфавита A, либо символ ”-”, который называется **indel**.

Среди всех возможных парных выравниваний ищется выравнивание, оптимальное относительно некоторой оценочной функции, которая обычно отражает биологический смысл выравнивания.

Множественное выравнивание определяется по аналогии с парным выравниванием как матрица $R = ((ri, j))$ размерности $M \times L$, где M — число выравниваемых последовательностей. Метод динамического программирования позволяет найти оптимальное парное выравнивание за приемлемое время. Но для множественного выравнивания многомерный вариант динамического программирования имеет экспоненциальную временную сложность относительно числа выравниваемых последовательностей[1], поэтому используются эвристические алгоритмы, не гарантирующие точность решения, но имеющие меньшую сложность и дающие приемлемое решение с точки зрения биологии.

Алгоритм работы ClustalW2

Алгоритм множественного выравнивания в ClustalW2 состоит из 4 шагов. Первый шаг рассчитывает матрицу расстояний, производя попарные выравнивания между всеми входными последовательностями. Стоит отметить, что в ClustalW2 существует два метода попарного выравнивания: быстрый, но дающий примерный результат - Fast Pairwise Alignment и точный, но медленный - Full Pairwise Alignment. Шаг номер два, строит бинарное дерево кластеров по полученной матрице расстояний. Третий шаг формирует по полученному дереву выравнивание, строящееся как прогрессивное множественное выравнивание по профилям, задаваемым деревом. В заключении используется итеративный процесс для улучшения получаемого выравнивания.

Как говорилось ранее, в параметрах программы ClustalW2 можно выбрать один из двух методов попарного выравнивания. В работе используется Full Pairwise Alignment, ввиду его большей практической пользы[3]. Full Pairwise Alignment использует алгоритм динамического программирования, который состоит из трех шагов. Сначала, выполняется прямой ход локального выравнивания по алгоритму Смита-Ватермана (Smith - Watermann), от начала обеих последовательностей до их конца, чтобы найти позицию (se1,se2) с максимальным количеством очков. Далее, выполняется обратный ход из позиции (se1,se2) к началу последовательностей для поиска другой позиции (sb1, sb2) с максимальным количеством очков, равным предыдущей позиции. В заключении, применяется алгоритм Майерса и Миллера (Myers & Miller) для подсчета штрафов за пропуски.

Сложность каждого из шагов алгоритма представлена в Таблице 1, где n - количество последовательностей, l_{ave} - средняя длина последовательности. Более подробно в статье[3].

Таблица 1: Сложность каждого шага алгоритма ClustalW2

Шаг	Сложность
Матрица расстояний	$O(n^2 l_{ave}^2)$
Бинарное дерево кластеров	$O(n^3)$
Прогрессивное множественное выравнивание (каждая итерация)	$O(n l_{ave} + l_{ave}^2)$
Прогрессивное множественное выравнивание (все итерации)	$O(n l_{ave}^2 + n^2 l_{ave})$
Общая сложность	$O(n^2 l_{ave}^2 + n^3)$

Поиск возможностей распараллеливания алгоритма ClustalW2

Для того чтобы найти «узкое место» алгоритма ClustalW2 была использована утилита программы *valgrind* - профайлер *callgrind*, для этого ClustalW2 был скомпилирован компилятором *gcc* с ключом **–g3** и запущен с помощью команды: **valgrind –tool=callgrind clustalw2 <файл с последовательностями>**

Результаты работы *valgrind* представлены в Таблице 2.

Таблица 2: Результаты профилирования кода. Время в секундах.

Название файла последовательности	Количество последовательностей	Средняя длина последовательности	Матрица расстояний	Бинарное дерево кластеров	Прогрессивное выравнивание
nucleotide_baliBase2	23	1399	12	6	2
nucleotide_baliBase1	28	1124	9	5	3
PfamTest.fasta	37	431	4	1	1
trans_baliBase4	86	676	40	26	8
trans_baliBase3	113	497	38	28	8
trans_baliBase1	116	486	41	27	6
trans_baliBase2	142	424	39	19	8
OxBench2	150	584	103	61	20
OxBench1	303	494	283	160	99

Из результатов профилирования кода видно, что больше всего времени программа проводит на первом этапе алгоритма множественного выравнивания – попарном выравнивании.

Все три шага алгоритма могут выполняться независимо для разных пар последовательностей, следовательно, можно организовать параллельную работу первой части алгоритма ClustalW2.

Параллельная реализация

Чтобы написать параллельную реализацию алгоритма полного попарного выравнивания требуется найти и исключить зависимости между переменными в коде. В результате анализа кода класса *Full Pairwise Alignment* была построена диаграмма зависимостей по переменным и функциям (см. Рис. 1).

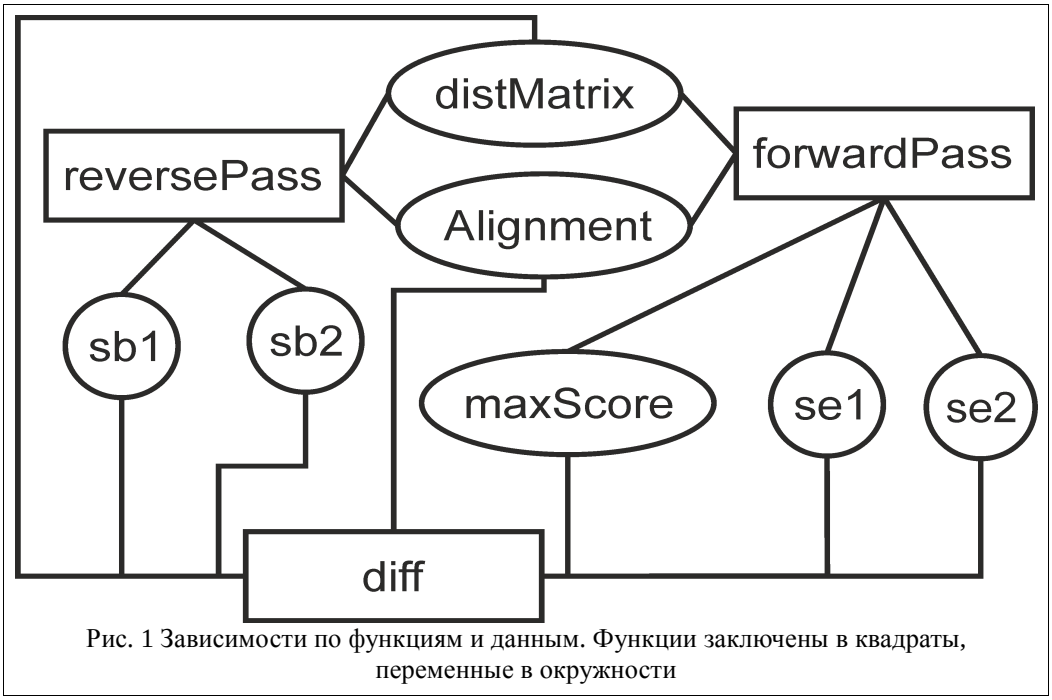

Рис. 1 Зависимости по функциям и данным. Функции заключены в квадраты, переменные в окружности

После выявления зависимостей в коде, структура классов ClustalW2 была изменена. В отдельные классы были выделены алгоритм Смита - Ватермана и Майерса - Миллера, *SWAlgo* и *MMAlgo* соответственно. Был создан отдельный класс для общих переменных *ExtendData*.

Основной цикл функции *pairwiseAlignment*, отвечающей за попарное выравнивание всех входных последовательностей, был «распараллелен» с помощью технологии OpenMP. В ходе исследований параллельной реализации оказалось, что стандартная процедура выделения памяти в glibc требует блокирования нитей в момент выделения памяти. Проблема была решена подключением библиотеки TCMalloc (http://goog-perftools.sourceforge.net/doc/tcmalloc.html).

В результате вышеописанных модификаций (см. Рис. 2) удалось проводить парные выравнивания независимо друг от друга.

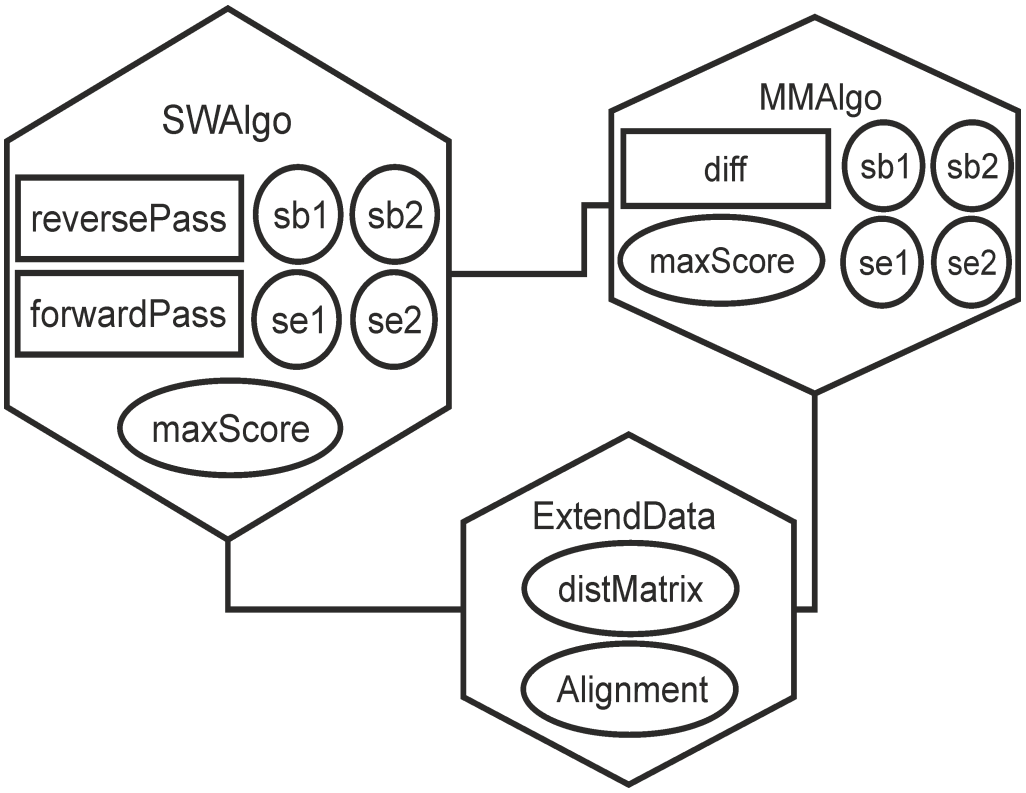

Рис. 2 Зависимости после проведенного рефакторинга. Функции заключены в квадраты, переменные в окружности, а классы в многогранники.

Методика тестирования

Для тестирования скорости и корректности работы параллельной реализации алгоритма ClustalW2 были написаны shell-скрипты. Скрипт *clustal_time_measure.sh* запускает параллельную реализацию ClustalW2 и замеряет время её работы. Скрипт *clustal_verify.sh* с помощью unix команды *diff* сверяет результаты работы параллельной реализации ClustalW2 с эталонными результатами программы на различных тестовых данных, описанных ниже.

Для проведения исследования было сформировано 9 файлов в FASTA формате, данные для последовательностей взяты из баз данных Pfam, oxBench и baliBase. Характеристики последовательностей представлены в Таблице 2.

Результаты тестирования

Параллельная реализация алгоритма ClustalW2, дает выигрыш в производительности примерно в 2 раза. Тесты были проведены на всех перечисленных в Таблице 2 файлах последовательностей. Результаты тестов представлены в Таблице 3. Тесты проводились на компьютере под управлением ОС Linux с процессором Intel Core i7 2630QM@2.00 ГГц имеющим 4 физических ядра и 4 Гб ОЗУ, число нитей в OpenMP установлено равным 4.

Таблица 3: Результаты тестирования

Название последовательности	Оригинальная версия ClustalW2 в секундах	Параллельная версия ClustalW2 в секундах	Ускорение
nucleotide_baliBase1	17	10	1,70
nucleotide_baliBase2	21	11	1,91
PfamTest.fasta	6	3	2,00
trans_baliBase2	66	29	2,28
OxBench2	184	79	2,33
trans_baliBase1	74	31	2,39
trans_baliBase3	74	30	2,47
trans_baliBase4	74	30	2,47
OxBench1	542	217	2,50

Заключение и дальнейшие планы

В результате проведенного исследования, можно сделать вывод о хорошем потенциале параллелизма первого этапа алгоритма ClustalW2. Поэтому можно попробовать реализовать его на массивно-параллельном устройстве, например GPU, но для этого требуется найти достойную не рекурсивную альтернативу алгоритму Майерса-Миллера. Также рассмотрев различия в исходном коде между ClustalW и ClustalW2, можно попробовать создать параллельную версию ClustalW2 используя MPI, взяв за основу статью Ли [4]. Анализ второго и третьего шагов алгоритма ClustalW2 – часть будущей работы.

Исходный код параллельной реализации ClustalW2 выложен по адресу http://code.google.com/p/parallel-clustalw2/.

ЛИТЕРАТУРА:

1. Humberto Carrillo, David Lipman "The Multiple Sequence Alignment Problem in Biology" // SIAM Journal on Applied Mathematics Vol. 48, No. 5 (Oct., 1988), pp. 1073-1082
2. K. Romanenkov, A. Salnikov "Comparison of quality and performance of parallel algorithms for multiple sequence alignment" proceedings of Moscow Conference on Computational Molecular Biology, 2011
3. Yongchao Liu, Bertil Schmidt, Douglas Maskell (2009) "Parallel Reconstruction of Neighbor-Joining Trees for Large Multiple Sequence Alignments using CUDA". In *IEEE International Workshop on High Performance Computational Biology (HiCOMB 2009)*. 1–8. IEEE Computer Society. Washington, DC, USA.
4. Kuo-Bin Li "ClustalW-MPI: ClustalW analysis using distributed and parallel computing" // Bioinformatics (2003) 19 (12):1585-1586.