УДК 004.942 DOI: 10.14529/cmse160103

ПАРАЛЛЕЛЬНЫЙ МЕТОД ОБЪЕДИНЕНИЯ РЕЗУЛЬТАТОВ РАБОТЫ ПРОГРАММ ПО СБОРКЕ ГЕНОМА[1]

К.В. Романенков, А.Н. Сальников, А.В. Алексеевский

В данной работе проводится исследование в области применения многопроцессорных систем для задачи коррекции сборки генома. Существует большое количество алгоритмических подходов к проблеме сборки генома из набора коротких фрагментов, при этом результаты их работы на одних и тех же экспериментальных данных зачастую существенно разнятся. Вследствие большого объема данных необходима организация вычислений в модели распределенной памяти на вычислительном кластере. Авторами предложен алгоритм объединения результатов работ геномных сборщиков, основанный на построении распределенного взвешенного графа контигов. Предлагаемый подход использует комбинацию выводов программ сборки геномов, что позволяет уменьшить фрагментированность контигов в результирующем наборе. Последовательная версия алгоритма реализована на C/C++ и доступна по адресу: https://bitbucket.org/kromanenkov/gar.

Ключевые слова: биоинформатика, многопроцессорные системы, параллельные алгоритмы.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Романенков К.В., Сальников А.Н., Алексеевский А.В. Параллельный метод объединения результатов работы программ по сборке генома // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 1. С. 24–34. DOI: 10.14529/cmse160103.

Введение

Современные высокотехнологичные автоматические методы расшифровки последовательностей ДНК (секвенирования) позволяют в течение короткого времени (несколько дней) и сравнительно недорого (несколько тысяч долларов) получить сотни миллиардов коротких последовательностей из четырех букв A, T, G, C, полученных прочтением фрагментов входного образца ДНК одного или нескольких организмов.

Расшифровка индивидуальных геномов людей и множества видов организмов, от вирусов и бактерий до сельскохозяйственных и домашних животных, существенно ускоряет прогресс в персонифицированной медицине, биоинженерии и биотехнологии, теории эволюции и других областях биологии [1].

Технология секвенирования заключается в следующем: в геноме выбирается случайный фрагмент, а затем происходит считывание двух последовательностей с его концов, причем центральная часть фрагмента остается непрочитанной. Эти последовательности называются парными чтениями(ридами), причем длина непрочитанной части фрагмента известна только приблизительно. В зависимости от технологии можно получать парные чтения длиной 100—250 нуклеотидов.

В идеале результатом сборки генома должны быть последовательности хромосом изучаемого организма; в случае человека — 24. Однако на пути от десятков миллиардов последовательностей длины 100—250 до нескольких последовательностей, представляющих

[1]Статья рекомендована к публикации программным комитетом Международной суперкомпьютерной конференции «Параллельные вычислительные технологии — 2015».

полный геном, возникают серьезные технологические, алгоритмические и вычислительные проблемы [1]. Полностью преодолеть возникающие трудности для больших геномов не удается или удается с большим трудом. Так, последняя сборка *hg38* генома человека состоит из 735 «скэффолдов» — непрерывных последовательностей, — вместо 24-х в идеале. Число нерасшифрованных нуклеотидов оценивается в 160 млн. (расшифрованных — 3 млрд 209 млн 286 тыс 105) [2]. Показательно также, что среди геномов 1542 видов эукариот, заявленных в БД NCBI в разделе секвенированных геномов, лишь 14 отнесены к группе полностью секвенированных геномов [3].

В алгоритмах ассемблирования (сборки геномов из коротких ридов) должны учитываться такие факторы, как неравномерность покрытия (число чтений, содержащих тот или иной нуклеотид генома) геномов ридами, которые по технологии получаются из случайных фрагментов ДНК; возможность и частоту ошибок в ридах; возможность наличия химерных ридов, составленных из разных частей ДНК; наличие в геномах длинных повторов, которые могут приводить к невозможности восстановление полной последовательности даже теоретически.

Наиболее сложной является сборка генома *de novo*. Задача сборки при наличии образца, например, сборки генома индивидуального человека при наличии референсного генома (секвенированный, собранный и проаннотированный геном организма того же вида, к которому относится анализируемый образец) более простая. Предложены десятки алгоритмов сборки de novo. Большинство из них основаны на построении графа де Брюйна и нахождении Эйлерового пути в нем. Однако из-за разных эвристик, заложенных на разных этапах сборки, результаты применения сборщиков существенно отличаются [4], см. также наши результаты далее в тексте. Важным обстоятельством, усложняющим сравнение алгоритмов, является то, что отсутствуют универсальные метрики оценки качества сборки. Причина заключается в том, что нет количественной оценки ошибок разного типа. Например, что лучше: большее число длинных контигов (однозначно расшифрованных непрерывных последовательностей) и скэффолдов (нескольких контигов, склеенных в единую последовательность с возможными пропусками или недостоверно определенными нуклеотидами между ними) либо уменьшение числа химер: контигов и скэффолдов, ошибочно склеенных из фрагментов разных хромосом?

Большой разрыв между желаемым результатом сборки *de novo* и получаемыми результатами, высокая актуальность создания более качественных сборщиков геномов для медицины и биологии ставят эту задачу в первый ряд актуальных вычислительных задач.

Статья организована следующим образом. В разделе 1 описываются особенности задачи коррекции сборок и разбираются существующие программные средства для решения этой задачи. В разделе 2 содержится краткое описание используемых данных и сборщиков. Раздел 3 описывает предложенный параллельный метод объединения результатов работ геномных сборщиков. В разделе 4 приведены результаты вычислительного эксперимента. В заключении анализируется оправданность использования вычислительных кластеров и описываются направления будущих исследований.

1. Задача коррекции сборки

Сборки генома, построенные сборщиками на основе ридов длины 100—150 нуклеотидов фрагментированы и содержат ошибки. Обычная практика при решении задачи сборки генома заключается в запуске нескольких сборщиков с различными параметрами, а затем

выбору наилучшего варианта согласно некоторым соображениям. Однако недавние исследования показывают, что достаточно распространена ситуация, при которой одни сборщики показывают лучший результат по одному из статистических критериев в сравнении с другими программами, и уступают им же по другому критерию [5]. С большой уверенностью можно утверждать, что для каждого сборщика существуют фрагменты в геноме, с задачей сборки которых он справляется лучше остальных. В основном, это объясняется деталями реализации стратегии того или иного сборщика по разрешению «проблемных» участков в геноме, например, связанных с повторами.

Согласуясь с этим предположением, в последнее время были созданы программные средства, реализующие стратегию согласования для пары результатов работы геномных сборщиков. Обычно один набор контигов считается «ведущим», а второй «ведомым», задача согласования заключается в объединении контигов из пары наборов, при этом необходимо обнаружить и изолировать проблемные регионы. Цель подобной стратегии — уменьшение фрагментированности наборов контигов и увеличение средней длины последовательностей в объединении. Основные проблемы, возникающие при согласовании результатов различных геномных сборщиков, — это ошибочное склеивание контигов, соответствующих различным фрагментам референсной последовательности (генома образца) и появление дубликатов в сборке. Вторая проблема возникает из-за того, что большая часть собираемого генома дублируется в наборах контигов, поэтому необходимо отфильтровывать последовательности, которые полностью покрываются либо непосредственно контигами, либо их объединениями.

Одной из первых программ для объединения набора контигов от разных сборщиков стала Reconciliator [6]. В ней производится поиск участков, являющихся уникальными как для ведомой, так и для ведущей последовательности. На следующем этапе закрываются пропуски в контигах из первого набора с использованием последовательностей из второго набора. В случае наличия нескольких вариантов выбирается тот, который отвечает лучшему статистическому критерию. Похожий подход использует GAA [5], строящий граф соответствия между наборами контигов, используемый для объединения сборок, и ZORRO [7], который предваряет шаг объединения этапом фильтрования контигов, содержащих ошибки. Программа GAM-NGS [4] ищет в контигах блоки соответствия, которые зависят от количества ридов, картирующихся на сравниваемые последовательности — таким образом, не проводят процедуру выравнивания каждого контига с каждым. Исходя из информации, полученной на стадии картирования ридов, строится граф сборок, анализируя который, можно установить участки несоответствия между наборами контигов.

Процесс объединения результатов работ геномных сборщиков является ресурсоемкой задачей. Время работы программы Reconciliator на одном из входных наборов данных составляет 24 часа, потребление оперативной памяти при этом составляет больше 100 Гб [6]. Все описанные программные продукты являются последовательными, либо работают в контексте модели общей памяти, что ограничивает их масштабируемость.

Вышеперечисленные программы объединения результатов работы геномных сборщиков предназначены для сопоставления только двух наборов контигов. Описываемый метод способен объединять контиги, полученные от произвольного числа сборщиков.

2. Описание используемых данных и сборщиков

2.1. Encephalitozoon cuniculi fungus

Для тестирования предложенного метода был выбран геном гриба Encephalitozoon cuniculi, содержащий 11 хромосом, общая длина которых составляет около 2,5 миллионов нуклеотидов. Парные риды длиной 100 символов, полученные в результате секвенирования генома данного организма, доступны на сайте Европейского биоинформатического института [8], референс, состоящий из 11 последовательностей хромосом, опубликован на сайте NCBI [9]. Из парных ридов нижеперечисленными сборщиками были получены последовательности контигов, которые и служили входными данными для описываемого метода. Наличие референсной последовательности для данного набора данных позволяет оценить корректность полученных сборок, поэтому параметры для запуска сборщиков подбирались таким образом, чтобы полученные ими результаты максимально соответствовали референсу. На рис. 1 показан пример картирования набора контигов, полученного от разных сборщиков на референсную последовательность, построенный с помощью программы QUAST [10].

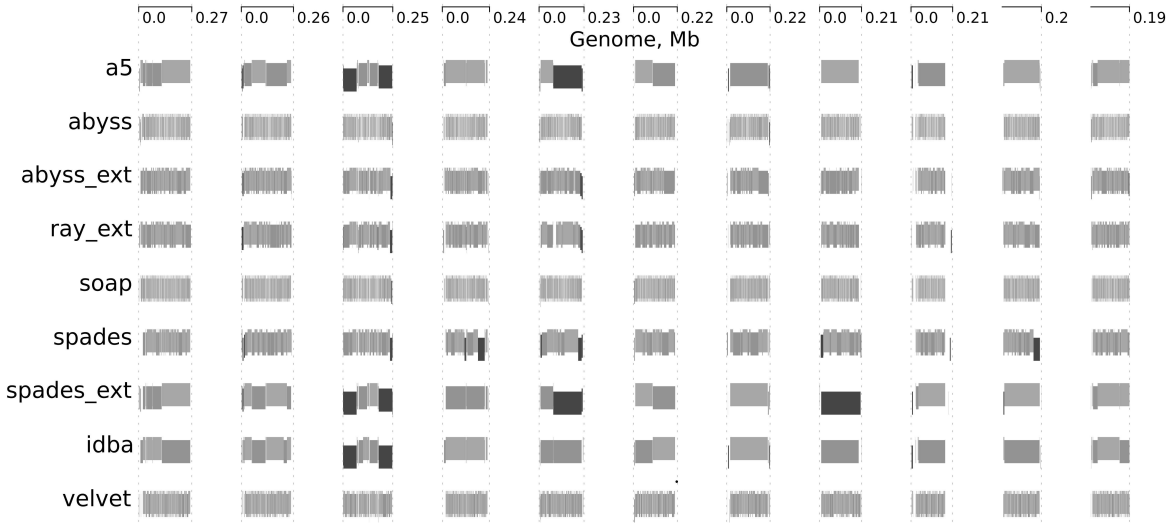

Рис. 1. Сравнение контигов, полученных от различных сборщиков, и референсных последовательностей

На горизонтальной оси располагаются результаты картирования наборов контигов на 11 хромосом референсной последовательности. Вдоль вертикальной оси расположены наборы контигов, полученные от разных сборщиков или от одного сборщика, но с разными параметрами. Участки, выделенные темным цветом, соответствуют ошибкам сборки (например, перестановке участков генома местами или объединению участков генома, которые в референсной последовательности не следуют друг за другом). Выравнивание контига и референсной последовательности изображается на рисунке сплошным прямоугольником соответствующей длины; таким образом, если в результате работы сборщика получилось много коротких контигов, это выражается в сильной фрагментированности диаграммы выравнивания.

2.2. Используемые сборщики

Ниже перечислены геномные сборщики, результаты работы которых объединяет предложенный метод. Все они используют концепцию графа де Брюйна и фигурируют в научных работах последнего времени, посвященных сравнению качества работы геномных сборщиков [11].

A5 — автоматически выбирает параметры сборки, внутри использует сборщик IDBA.

ABYSS — собирает геном в модели распределенной памяти, используя библиотеку MPI. Для определения участков перекрытий использует распределенную хеш-таблицу.

IDBA — итеративно изменяет параметры сборки, на каждой итерации дополняя риды контигами, полученными на предыдущей итерации.

RAY — может собирать геном в модели распределенной памяти, используя библиотеку MPI. Использует оригинальную эвристику для выявления повторов в геноме.

SOAPdenovo — использует разреженную хеш-таблицу для экономии оперативной памяти, что может приводить к ошибкам сборки. Для этого сборщика характерно относительно малое время работы.

SPAdes — использует различные параметры сборки для участков генома с различной глубиной покрытия.

Velvet — один из первых и самых распространенных сборщиков для коротких ридов. Необходимо вручную задавать большой набор параметров сборки.

3. Описание метода объединения

Основная идея алгоритма заключается в построении распределенного взвешенного графа контигов, используемого для этапа согласования последовательностей. Каждая вершина графа соответствует контигу, а ребро — ситуации концевого перекрытия контигов, то есть случаю, когда конец одной последовательности идентичен или практически идентичен началу другой последовательности (в данной работе последовательности считались перекрывающимися, если количество идентичных символов в выравнивании $\geq$90% от длины перекрытия). Также в вершина сохраняется информация о наборе, к которому принадлежал соответствующей ее контиг. Для поиска перекрытий выполняется попарное выравнивание контигов из каждой пары наборов. Выравнивание последовательностей производится программой MEGABLAST, которая является вариацией программы BLAST, отличающейся от нее более высокой скоростью работы и ориентацией на поиск участков сходства с малым числом неточных совпадений. В силу того, что каждый контиг будет многократно участвовать в выравнивании, все множества контигов предварительно индексируются.

Программная реализация выполнена на языке C/C++, для организации параллельных вычислений и пересылок данных между процессами используется библиотека MPI. Каждый MPI-процесс хранит в памяти часть вершин и ребер взвешенного графа контигов.

Общая схема предлагаемого метода выглядит следующим образом:

1. Считывание контигов из файлов, полученных в результате работы геномных сборщиков. Распределение контигов по процессам, сбор статистики: максимум, минимум, медиана длины контигов и т.д.
2. Каждый MPI-процесс для каждого своего контига производит выравнивание программой MEGABLAST со всем множеством контигов на всех процессорах. Цель парного выравнивания последовательностей — расположить символы последовательностей друг

под другом, вставляя специальные "пробельные"символы таким образом, чтобы наиболее похожие фрагменты одной последовательности выровнялись относительно аналогичных фрагментов другой последовательности. Помимо непосредственного расположения последовательностей программа MEGABLAST собирает статистику выравнивания: статистическая значимость выравнивания, количество "пробельных"символов, количество идентичных позиций в выравнивании и т.д. Эта информация сохраняется и будет использована позже при построении взвешенного графа контигов. Каждый процесс строит матрицу с результатами выравнивания. При этом каждый процесс запрашивает у остальных процессов последовательности контигов, а в ответ отправляет номера своих контигов и полученные им результаты выравнивания.

3. Построение распределенного взвешенного графа контигов. Вес ребра, соединяющего пару вершин, соответствующих контигам, определяется, исходя из результата парного выравнивания контигов, и характеризует «перспективность» объединения конкретной пары последовательностей. В формулу для расчета веса ребра входят статистическая значимость выравнивания, количество идентичных позиций в выравнивании, длины последовательностей и другие параметры со своими весовыми коэффициентами. На этапе объединения контигов предпочтение отдается ребрам с большим весом. Каждый процесс владеет информацией о распределении вершин и ребер графа по процессорам.
4. Кластеризация построенного графа с помощью LabelRank-подобного алгоритма на непересекающиеся области так, чтобы максимизировать вес внутри каждой области и минимизировать вес ребер между областями. Процесс разбиения графа позволяет уменьшить число перекрытий, получающихся в результате ошибки сборки или из-за наличия в геноме длинных повторов. Данный подход является масштабируемым, что достигается отказом от хранения графа контигов или матрицы попарного выравнивания в общей памяти. Если ребро рассматриваемого контига находится в памяти другого процесса, то ему отправляется сообщение типа точка-точка, ответом на которое будет служить вес ребра. Предполагается, что каждая область будет содержать максимальное число участков, соответствующих одному фрагменту генома. Это объясняется следующим образом: предположим, существуют два контига C_i и C_j, перекрывающиеся из-за наличия у них общего повтора. Если при этом в геноме C_i не располагается рядом с C_j, то C_i будет иметь малое число перекрытий с контигами, перекрывающимися с C_j, и не попадет в область, к которой будет отнесен C_j. В связи с этим уменьшается вероятность склеивания пары контигов, перекрывающихся благодаря тому, что они оба содержат участок повтора.
5. В каждой области, содержащей достаточное большое количество последовательностей (в данной работе такими считались области, содержащие 5 и более последовательностей), выполняется объединение контигов жадным алгоритмом, в итоговый результат попадают последовательности длиннее порогового значения.

На рис. 2 показан пример работы алгоритма для 3 наборов контигов. Следует отметить, что если один контиг целиком содержится в другом, то ребро, соответствующее этой паре контигов, не попадает в граф, что отражено в иллюстрации.

На ранних этапах исследования основной метод предваряла операция картирования ридов на наборы контигов с помощью сторонней программы и выделение в контигах участков мало покрытых короткими чтениями. Такие участки могут сигнализировать о потенциальных ошибках сборки, поэтому они разделялись с целью уменьшения числа последователь-

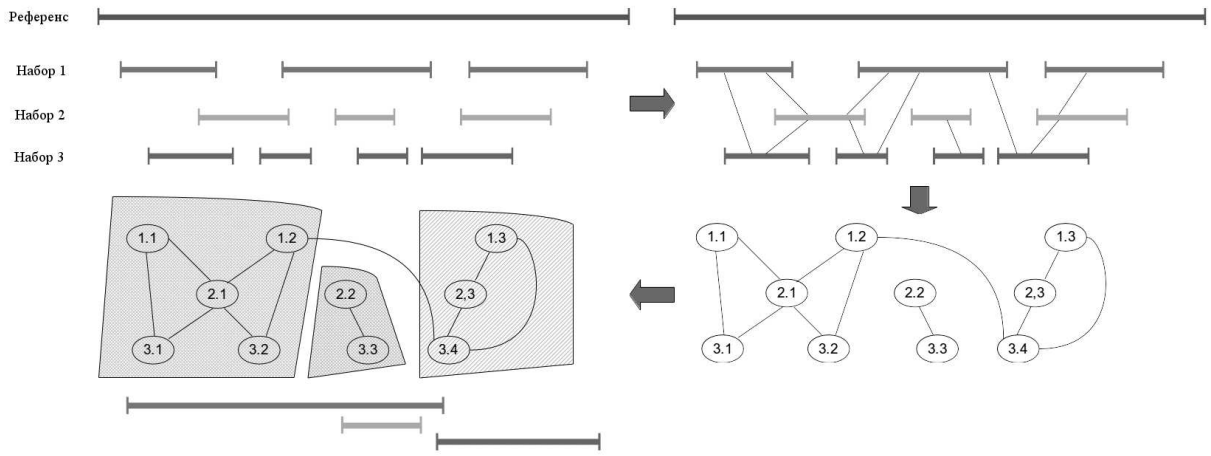

Рис. 2. Пример работы предложенного метода для 3 наборов контигов

ностей, содержащих участки, не следующих друг за другом в референсном геноме. Однако эксперименты показали, что эта операция не только не разделяла ошибочные последовательности, но и дробила корректно собранные контиги, в связи с чем из дальнейшего исследования этот этап был исключен.

Ввиду большой размерности матрицы попарного выравнивания (предположим, имеется 10 наборов по 10000 контигов, тогда необходимо $(10 * 9 * 10^4 * 10^4)/2$ структур для хранения результатов выравнивания; даже если размер структуры составит 1 байт, это потребует более 30 Гб памяти) она хранится в разреженном виде. Реализация разреженной матрицы и списков смежности графа выполнена с помощью хэш-таблицы.

4. Вычислительный эксперимент

Тестирование предложенного метода проводилось на суперкомпьютере «Ломоносов». Вначале были получены наборы контигов от разных сборщиков, а затем был запущен процесс объединения последовательностей. В таблице представлены результаты работы метода, для оценки результата использованы следующие метрики: число ошибок сборки (перестановка участков генома местами; объединение участков генома, которые в референсной последовательности не следуют друг за другом), общее число контигов, N50 и длиннейший непрерывный участок (ДНУ), картирующийся на референс. Метрики числа ошибок сборки и длиннейшего непрерывного участка являются референс-зависимыми, для подсчета числа контигов и N50 референсная последовательность не требуется.

Метрика N50 определяет величину, при которой контиги длиннее значения этой величины составляют половину собранного генома, и считается стандартом сравнения геномных сборок в случае отсутствия референса. Для подсчета величины длиннейшего непрерывного участка сборки используется следующая схема: контиги выравниваются на референсный геном, при этом если контиг полностью не картируется на референсную последовательность (например, переставлены местами фрагменты контига или контиг содержит неверный кусок), то он разбивается на более мелкие участки, целиком картирующиеся на геном. Длина самого протяженного полученного участка и считается величиной ДНУ.

В работах, посвященных анализу качества геномных сборок, обычно считается, что более качественная сборка характеризуется меньшим числом контигов, меньшим числом

ошибок сборки, большим значением N50 и большим значением длиннейшего непрерывного участка.

Таблица

Результаты работы по объединению набора контигов

Сборщик	Ошибки сборки	Число контигов	N50	ДНУ
A5	14	16774	1401	199363
Abyss	10	30933	9076	75379
Ray	9	15087	10177	54559
SOAPdenovo	2	6340	1558	6989
Spades	8	42236	1883	199986
IDBA	9	1533	65715	199363
Velvet	3	3693	3344	17502
Предложенный метод	7	1944	50631	199363

Из таблицы видно, что в процессе объединения результатов часть контигов была отфильтрована, что объясняется либо их малой длиной, либо малым количеством перекрытий с другими последовательностями. В объединение попал контиг, содержащий длиннейший непрерывный участок из наборов A5 и IDBA, одновременно с этим значительно уменьшилось число контигов, что впрочем не привело к ощутимому увеличению числа ошибок.

Заключение

Полученные результаты позволяют утверждать, что предложенный метод справился с основной задачей согласования геномных сборок: уменьшение фрагментации последовательностей и увеличение средней длины последовательности в объединении.

Авторы хотят отметить исключительную удачность решения использовать вычислительный кластер для объединения сборок. Это позволит в дальнейшем адаптировать предложенный метод для более длинных геномов, например генома человека, и большего числа сборщиков. В рамках дальнейшего исследования планируются следующие действия:

1) проверка работы предложенного подхода для объединения сборок геномов, размер которых составляет около 1 Гб;

2) изучение применимости графовых алгоритмов кластеризации для выделения областей графа, соответствующих контигам, полученных из одной хромосомы;

3) изучение возможности хранения контигов в сжатом виде в оперативной памяти (например, используя FM-индекс [12]) для экономии ОП и возможности работы с большими геномами;

Работа частично поддержана грантом РНФ №14-50-00029.

Литература

1. Miller J.R., Koren S., Sutton G. Assembly algorithms for next-generation sequencing data // Genomics. 2010. Vol. 95, No. 6. P. 315–327. DOI: 10.1016/j.ygeno.2010.03.001.
2. NCBI, БД Assembly, геном человека. URL: `http://www.ncbi.nlm.nih.gov/assembly/883148` (дата обращения: 1.08.2015).
3. Метаинформация о геномах эукариотов на сайте NCBI. URL: `ftp://ftp.ncbi.nlm.nih.gov/genomes/GENOME_REPORTS/eukaryotes.txt` (дата обращения: 1.08.2015).
4. Vicedomini R., Vezzi F., Scalabrin S., Arvestad L., Policriti A. GAM-NGS: genomic assemblies merger for next generation sequencing // BMC Bioinformatics. 2013. Vol. 14(Suppl.7), No. 1. P. 1–18. DOI: 10.1186/1471-2105-14-S7-S6.
5. Yao G., Ye L., Gao H., Min P., Warren W.C., Weinstock G.M. Graph accordance of next-generation sequence assemblies // Bioinformatics. 2012. Vol. 28, No. 1. P. 13–16. DOI: 10.1093/bioinformatics/btr588.
6. Zimin A.V., Smith D.R., Sutton G., Yorke J.A. Assembly reconciliation // Bioinformatics. 2008. Vol. 24, No. 1. P. 42–45. DOI: 10.1093/bioinformatics/btm542.
7. Zorro – The masked assembler. URL: `http://lge.ibi.unicamp.br/zorro/` (дата обращения: 22.07.2015).
8. European Nucleotide Archive. URL: `http://www.ebi.ac.uk/ena/data/view/SRR122309` (дата обращения: 1.08.2015).
9. Encephalitozoon cuniculi GB-M1. URL: `http://www.ncbi.nlm.nih.gov/genome/39?genome_assembly_id=22671` (дата обращения: 1.08.2015).
10. Gurevich A., Saveliev V., Vyahhi N., Tesler G. QUAST: quality assessment tool for genome assemblies // Bioinformatics. 2013. Vol. 29, No. 8. P. 1072–1075. DOI: 10.1093/bioinformatics/btt086.
11. Koren S., Treangen T.J., Hill C.M., Pop M., Phillippy A.M. Automated ensemble assembly and validation of microbial genomes // BMC Bioinformatics. 2014. Vol. 15, No. 5. P. 126–134. DOI: 10.1186/1471-2105-15-126.
12. Simpson J.T., Durbin R. Efficient construction of an assembly string graph using the FM-index // Bioinformatics. 2010. Vol. 26, No. 12. P. 367–373. DOI: 10.1093/bioinformatics/btq217.

Романенков Кирилл Владимирович, аспирант кафедры суперкомпьютеров и квантовой информатики факультета Вычислительной математики и кибернетики, Московский государственный университет (Москва, Российская Федерация), kromanenkov2@yandex.ru.

Сальников Алексей Николаевич, к.ф.-м.н., ведущий научный сотрудник факультета Вычислительной математики и кибернетики, Московский государственный университет (Москва, Российская Федерация), salnikov@cs.msu.ru.

Алексеевский Андрей Владимирович, к.ф.-м.н., ведущий научный сотрудник, Научно-исследовательский институт физико-химической биологии им. А.Н. Белозерского (Москва, Российская Федерация), aba@belozersky.msu.ru.

Поступила в редакцию 17 августа 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2016, vol. 5, no. 1, pp. 24–34

DOI: 10.14529/cmse160103

PARALLEL MERGING METHOD TO INTEGRATE DIFFERENT GENOME ASSEMBLIES

K.V. Romanenkov, Lomonosov Moscow State University, Moscow, Russian Federation
A.N. Salnikov, Lomonosov Moscow State University, Moscow, Russian Federation
A.V. Alexeevski, A.N. Belozersky Institute Of Physico-Chemical Biology, Moscow, Russian Federation

In this paper research in the field of application multiprocessor systems for genome assemblies reconciliation has been carried out. A large number of algorithmic approaches aimed to solve the task of *de novo* assembly from short reads, however the results of their work on the same raw data often differ essentially. Due to the large data volume the computations in the distributed memory model on computational cluster are required. Authors develop merging algorithm to integrate different genome assemblies based on distributed weighted contig graph. The proposed method integrates a combination of draft assemblies reducing resulting contigs fragmentation. Sequential version of the algorithm is implemented in C/C++ and is available at https://bitbucket.org/kromanenkov/gar.

Keywords: bioinformatics, multiprocessor systems, parallel algorithms.

FOR CITATION

Romanenkov K.V., Salnikov A.N., Alexeevski A.V. Parallel Merging Method to Integrate Different Genome Assemblies. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 1. pp. 24–34. (in Russian) DOI: 10.14529/cmse160103.

References

1. Miller J.R., Koren S., Sutton G. Assembly algorithms for next-generation sequencing data. Genomics. 2010. vol. 95, no. 6. pp. 315–327. DOI: 10.1016/j.ygeno.2010.03.001.
2. NCBI, Assembly database, human genome. URL: `http://www.ncbi.nlm.nih.gov/assembly/883148` (accessed: 1.08.2015).
3. The meta-information about the eukaryotes' genomes on NCBI site. URL: `ftp://ftp.ncbi.nlm.nih.gov/genomes/GENOME_REPORTS/eukaryotes.txt` (accessed: 1.08.2015).
4. Vicedomini R., Vezzi F., Scalabrin S., Arvestad L., Policriti A. GAM-NGS: genomic assemblies merger for next generation sequencing. BMC Bioinformatics. 2013. vol. 14(suppl.7), no. 1. pp. 1–18. DOI: 10.1186/1471-2105-14-S7-S6.
5. Yao G., Ye L., Gao H., Min P., Warren W.C., Weinstock G.M. Graph accordance of next-generation sequence assemblies. Bioinformatics. 2012. vol. 28, no. 1. pp. 13–16. DOI: 10.1093/bioinformatics/btr588.

6. Zimin V.A., Smith D.R., Sutton G., Yorke J.A. Assembly reconciliation. Bioinformatics. 2008. vol. 24, no. 1. pp. 42–45. DOI: 10.1093/bioinformatics/btm542.

7. Zorro – The masked assembler. URL: `http://lge.ibi.unicamp.br/zorro/` (accessed: 22.07.2015).

8. European Nucleotide Archive. URL: `http://www.ebi.ac.uk/ena/data/view/SRR122309` (accessed 1.08.2015).

9. Encephalitozoon cuniculi GB-M1. URL: `http://www.ncbi.nlm.nih.gov/genome/39?genome_assembly_id=22671` (accessed: 1.08.2015).

10. Gurevich A., Saveliev V., Vyahhi N., Tesler G. QUAST: quality assessment tool for genome assemblies. Bioinformatics. 2013. vol. 29, no. 8. pp. 1072–1075. DOI: 10.1093/bioinformatics/btt086.

11. Koren S., Treangen T.J., Hill C.M., Pop M., Phillippy A.M. Automated ensemble assembly and validation of microbial genomes. BMC Bioinformatics. 2014. vol. 15, no. 5. pp. 126–134. DOI: 10.1186/1471-2105-15-126.

12. Simpson J.T., Durbin R. Efficient construction of an assembly string graph using the FM-index. Bioinformatics. 2010. vol. 26, no. 12. pp. 367–373. DOI: 10.1093/bioinformatics/btq217.

Received August 17, 2015.