АЛГОРИТМ ВЫРАВНИВАНИЯ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДНК ДЛЯ МОДЕЛИ MAPREDUCE

Славнейшев Ф.В.

магистрант кафедры КТ факультета ИТиП НИУ ИТМО,
slavnejshev@mail.ru

Аннотация: Выравнивание последовательностей ДНК является важной и одновременно вычислительно сложной задачей биоинформатики. В данной работе представляется алгоритм выравнивания последовательностей ДНК для популярной модели распределенных вычислений MapReduce. Также проводится анализ уже существующих решений для данной модели.

Введение

Выравнивание последовательностей ДНК является одной из важнейших задач в биоинформатике. Решение этой задачи требуется при филогенетическом анализе, определении функций отдельных частей ДНК на основе сравнения, нахождении эволюционно консервативных участков.

Однако решение задачи выравнивания последовательной ДНК требует больших вычислительных ресурсов. Например, в случае ДНК человека и при длине чтений в 100 символов производится выравнивание как минимум 600 миллионов чтений с допущением трех несовпадений относительно образца ДНК длиной три миллиарда символов [1]. Решение этой задачи на обычном персональном компьютере может потребовать несколько недель или даже месяцев вычислений. Применение распределенных вычислений и использование кластеров машин позволяет сильно сократить требуемое для выравнивания время.

Модель MapReduce

Одной из самых популярных моделей распределенных вычислений является MapReduce. Использование данной модели позволяет решать задачи, связанные с большими объемами данных. Еще одним плюсом данной модели является ее хорошая масштабируемость.

Работа MapReduce состоит из двух шагов:

1. map-шаг — производится обработка входных данных и формирование промежуточных результатов, которые объединяются по некоторому ключу (идентификатору решаемой задачи) в группы;
2. reduce-шаг — происходит обработка каждой группы промежуточных результатов и формируется окончательный ответ для решаемой задачи.

Из описания модели видно, что необходимым условием ее эффективной работы является разбиение решаемой задачи на множество небольших подзадач. В случае выравнивания последовательной ДНК данное условие можно легко удовлетворить, т. к. выравнивание каждого чтения фактически не зависит от других. Информация об образце может быть предоставлена каждому узлу в сети целиком, либо образец может быть также разбит на части.

Обзор имеющихся решений

Рассмотрим наиболее известные алгоритмы выравнивания последовательностей ДНК для модели MapReduce:

1. CloudBurst [2] — простой и быстрый при работе с небольшими объемами данных алгоритм, основанный на методе «seed and extend». Выполняется всего один прогон работы MapReduce, в ходе которого находятся общие k-меры (части последовательности ДНК длиной символов), а затем производится выравнивание чтения в каждой найденной позиции. Алгоритм подходит для работы с короткими чтениями (36 пар оснований) и небольшим по длине образцом. Для более длинных чтений время работы и требуемая память жесткого диска сильно возрастают;
2. BlastReduce [3] — распределенная версия алгоритма BLAST. Главным отличием данного алгоритма от CloudBurst является процесс объединения общих k-меров для уменьшения числа выполняемых выравниваний. Выполняется три прогона MapReduce. Однако BlastReduce имеет примерно такие же ограничения на длину чтений и образца, что и CloudBurst;
3. Crossbow [4] — алгоритм нахождения однонуклеотидных полиморфизмов (SNP), на первом этапе которого выполняется выравнивание при помощи алгоритма Bowtie. Алгоритм Bowtie использует для выравнивания чтений созданный в результате препроцессинга образца BWT. На каждом узле запускается отдельная копия Bowtie, а в память узла загружается BWT, который может занимать несколько гигабайт оперативной памяти, что повышает требования к ЭВМ, на которых может быть запущен данный алгоритм. Еще одним минусом алгоритма является невозможность учета ошибок вставки и удаления, т. к. текущая версия Bowtie может работать лишь с ошибками замены.

Описание алгоритма

Основная идея алгоритма заключается в том, чтобы разбить каждое чтение на k-меры и определить их позиции в образце, т. е. найти общие k-меры. Затем по расположению отдельных частей можно найти наиболее вероят-

ную позицию исходного чтения относительно образца. После нахождения приблизительного положения чтения в образце можно произвести выравнивание оставшихся частей чтения при помощи существующих алгоритмов. Такой подход позволяет произвести выравнивание чтений, содержащих многочисленные несоответствия с образцом.

Рассмотрим более подробно, как при помощи модели MapReduce можно достичь требуемых результатов. Генерацию и нахождение позиций k-меров можно сделать за один прогон MapReduce. На map-шаге каждое чтение разбивается некоторым образом на k-меры, которые выдаются вместе с указанием позиции в чтении и идентификатора чтения в качестве промежуточных результатов. Стоит отметить, что позицию k-мера задает не только смещение от начала чтения до начала k-мера, но и указание того, из прямого или обратно-комплементарного варианта чтения взят этот k-мер. K-мер используется в качестве ключа. При этом на часть mapper'ов подается разбитый на части образец, который обрабатывается аналогичным образом. В итоге каждый reducer получает на вход k-мер и множество позиций, в которых он встречается. Если k-мер не встречается в чтениях или в образце, то reducer пропускает такой k-мер. В противном случае reducer в качестве ответа выдает множество позиций k-мера. Стоит отметить, что в отличие от алгоритмов CloudBurst и BlastReduce вместе с k-мерами не выводятся сами чтения и части образца. При больших длинах чтений это позволяет избежать генерации огромных объемов побочных данных.

На следующем этапе определяется положение каждого чтения по позициям его частей. На map-шаге обрабатываемое множество позиций разбивается на два — множество позиций в чтениях и множество позиций в образце. Затем для каждого встретившегося ID чтения выводится множество позиций в образце, дополненное позицией в чтении. В качестве ключа используется ID чтения. На reduce-шаге для каждого чтения имеется некоторое множество позиций в нем, на которых лежат общие k-меры. При этом для каждого такого k-мера может быть несколько вариантов расположения в образце. Каждый k-мер покрывает некоторую часть образца. Задача reducer'а — объединить все покрытия в группы. Для этого для каждого покрытия вычисляется позиция, в которой находилось бы чтение, если бы все чтение до начала k-мера полностью совпадало с образцом. Производится объединение близких по этому параметру покрытий «жадным» алгоритмом.

На заключительном этапе работы алгоритма производится окончательное выравнивание чтений. Каждое имеющееся покрытие дополнено чтением. Для того, чтобы произвести выравнивание, нужно иметь ту часть образца, в которой по предположению находится чтение. На map-шаге образец разбивается на блоки, каждый из которых выводится с ключом, равным номеру блока. Все имеющиеся сгруппированные покрытия выводятся с ключом, сформированным по тому же правилу. На reduce-шаге имеется блок из образца и множество чтений с покрытиями, для каждого из которых произ-

водится выравнивание непокрытых частей чтения. Для этого используется алгоритм Нидлмана-Вунша. Если расстояние Левенштейна между чтением и образцом превышает указанный в параметрах алгоритма максимум, выравнивание отбрасывается. Если требуется для каждого чтения найти максимум одно выравнивание, возможен еще один дополнительный этап, на котором для каждого чтения находится лучшее выравнивание.

Текущие результаты

Алгоритм имеет определенные ограничения в использовании. При слишком малом значении длины k-меров на втором этапе генерируется огромное число покрытий, которые могут потребовать больших затрат памяти жесткого диска. При этом на значение длины налагается ограничение параметром (максимальное расстояние Левенштейна между чтением и образцом) следующего вида:

$$k \leqslant \frac{Len_{read}}{d+1},$$

где Len_{read} — длина чтения. Таким образом, для некоторых значений параметра d и при определенных длинах чтения и образца выравнивание данным алгоритмом осуществить затруднительно.

Производился тестовый запуск алгоритма на машине с четырехъядерным процессором Core i5 2.80 GHz и четырьмя гигабайтами оперативной памяти. В качестве образца использовалось ДНК Escherichia coli, размер которой примерно четыре миллиона пар оснований. Производилось выравнивание четырех миллионов чтений длиной 36 пар оснований каждое. Для каждого чтения находилось лучшее выравнивание, то есть производился и дополнительный четвертый этап. На каждом этапе запускалось два mapper'а и один reducer.

В результате удалось произвести выравнивание 95,431 % чтений за 8 минут при $d=1$ и $k=18$. При $d=2$ и $k=12$ удалось произвести выравнивание 97,512 % чтений менее чем за 15 минут. Планируется произвести сравнение с уже имеющимися алгоритмами выравнивания как при работе с короткими, так и с длинными чтениями.

Заключение

В данной работе были рассмотрены существующие распределенные алгоритмы выравнивания последовательностей ДНК для модели MapReduce. Также был представлен трехэтапный алгоритм выравнивания последовательностей ДНК, при построении которого были учтены проблемы уже имеющихся алгоритмов. Разработанный алгоритм был протестирован на реальных данных и показал высокую скорость выравнивания чтений.

Литература

1. *Liu C.-M., Lam T.-W., Wong T., Wu E., Yiu S.-M., Li Z., Luo R., Wang B., Yu C., Chu X., Zhao K., Li R*. Soap3: GPU-Based Compressed Indexing and Ultra-Fast Parallel Alignment of Short Reads. Third Workshop Massive Data Algorithmics, June 2011.
2. *Schatz M.C.* CloudBurst: highly sensitive read mapping with MapReduce. [Электронный ресурс]. Режим доступа http://bioinformatics.oxfordjournals.org/content/25/11/1363.full.pdf свободный. Яз. англ. (дата обращения 19.04.13).
3. *Schatz M.C.* BlastReduce: High Performance Short Read Mapping with MapReduce. [Электронный ресурс]. Режим доступа http://www.cs.umd.edu/Grad/scholarlypapers/papers/MichaelSchatz.pdf свободный. Яз. англ. (дата обращения 19.04.13).
4. *Langmead B., Schatz M.C., Lin J., Pop M., Salzberg S.L.* Searching for SNPs with cloud computing. [Электронный ресурс]. Режим доступа http://genomebiology.com/content/pdf/gb-2009-10-11-r134.pdf свободный. Яз. англ. (дата обращения 19.04.13).