



ASSOCIATIVE COMPUTING IMPLEMENTATION LIBRARY CUSTAR: DATA REPRESENTATION FOR BIOINFORMATICS PROBLEMS

T. V. Snytnikova

Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russia

DOI: 10.24412/2073-0667-2023-1-60-68

EDN: QWFFMA

Over the past few years, genome processing has become a widely sought-after task. Both medical laboratories (from PCR tests to genetic passports) and research teams are engaged in various processing options. At the same time, both the first and the second process large amounts of data either due to the number of samples, or due to the length of these samples: from tens of thousands to several billion nucleotides. Note that a huge part of the calculations is related to the search for individual nucleotides or their sequences in a larger sequence or in a large number of sequences. So it is advisable to use associative parallel computing. But associative architectures are not represented on the computer hardware market, unlike widely available graphics accelerators. The cuSTAR library was designed to implement associative computing model STAR-machine on graphics accelerators. In this paper, a method of organizing data for processing genomes by associative algorithms is proposed.

In this paper, we propose several methods of data organization. Such an organization allows the use of associative algorithms to solve various tasks related to genome processing. Let's recall a brief description of the associative model of the STAR machine, and its cuSTAR implementation. Both the castor library and its STAR machine model use three types of data for associative processing. The Table type stores data as a binary table. The Slice type is used to access the bit column, and the word type is used to access the bit string. It should be noted that data processing is performed mainly using bit columns. Therefore, the presentation of data in the cuSTAR system is fundamentally different. Usually, a sequence of nucleotides is represented by a array of characters. It can be considered as a binary table in which the rows specify one character. That is, the data is stored line by line. To use cuSTAR, a variable of type Table is stored by columns.

The alphabet of nucleotides consists of the symbols A (adenine), C (cytosine), G (guanine) and T (thymine). Also, the “-” symbol is often used in the data to indicate possible gaps in reading, insertions or deletions in the nucleotide sequence. Thus, four or five characters are used, depending on the task. We propose two ways to encode a sequence of nucleotides. The first method is optimized for memory usage. The second method is optimized for the search time of the nucleotide in the sequence. The memory-optimized method uses the following encoding: “000” for “-” symbol, “001” for adenine, “011” for cytosine, “101” for guanine, “111” for thymine. The time-optimized method uses the following encoding: “1000” for adenine, “0100” for cytosine, “0010” for guanine, “0001” for thymine. It uses 4 bits instead of 3 bits, but allows you to replace the task of searching for a word in the table with a less time-consuming one. To find all occurrences of a nucleotide in the sequence, one needs to determine the position “1” in the code of this nucleotide. The proposed data encoding methods are more compact than the standard representation in the form of an array of characters. The time-optimized method makes it possible to search for nucleotides in a sequence an order of magnitude faster than the procedure from

the t memory-optimized method. But the memory-optimized method is preferable if the representation of the nucleotide sequence in the form of a graph is used. And in this case, the de Bruijn graph is constructed from the original sequence of nucleotides in a trivial way. Although with symbolic encoding of nucleotides, this is a time-consuming and memory-consuming task.

When using cuSTAR, it is easy to construct a de Bruijn graph from a sequence of nucleotides of any parameter k. The graph is given by a list of edges, which is one of the standard representation for associative processing. Note that by defining the graph as a list of edges, we avoid problems associated with repeating arcs.

When reading the sequence, a table GEN of size $3l$ is formed, where l is the length of the input sequence. For a graph given by a list of arcs, we form tables LEFT and RIGHT of size $3k(l - k)$. The table LEFT is obtained by copying k times the columns of the GEN into the corresponding columns with an upward shift. In turn, the table RIGHT is obtained by copying with a shift up one row of the table LEFT. Copying of all tables is performed in parallel.

Since genome processing involves multiple searches over a large amount of data, the development of associative algorithms for this area is relevant. The applied value of the work consists in the possibility of executing these algorithms on graphics accelerators — widespread equipment from personal computers to cluster systems.

Key words: associative parallel algorithms, bioinformatics, GPU, CUDA.

References

1. Snytnikova T. V., Nepomniaschaya A. Sh. Reshenie zadach na grafah s pomoshch'yu STAR-mashiny, realizuemoj na graficheskikh uskoritelyah // Prikladnaya diskretnaya matematika. 2016. Vol. 3 (33). P. 98–115.
2. Snytnikova T. V. Realizaciya modeli associativnyh vychislenij na gpu: biblioteka bazovyh procedur yazyka star. // Vychislitel'nye metody i programmirovaniye. Novye vychislitel'nye tekhnologii. 2018. Vol. 19. P. 85–95.
3. Compeau Ph., Pevzner P. A., Tesler G. How to apply de Bruijn graphs to genome assembly. // Nature Biotechnology. 2011. Vol. 29(11). P. 987–991.



БИБЛИОТЕКА РЕАЛИЗАЦИИ АССОЦИАТИВНЫХ ВЫЧИСЛЕНИЙ НА ГРАФИЧЕСКИХ УСКОРИТЕЛЯХ CUSTAR: ПРЕДСТАВЛЕНИЕ ДАННЫХ ДЛЯ ЗАДАЧ БИОИНФОРМАТИКИ

Т. В. Снытникова

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия

УДК 591.684

DOI: 10.24412/2073-0667-2023-1-60-68

EDN: QWFFMA

Система cuSTAR разработана для реализации ассоциативных алгоритмов на GPU. Известно, что ассоциативные вычисления дают преимущество в решении проблем, для которых характерны поисковые запросы по большому объему неструктурированных данных. К такой проблеме относится и обработка геномов. С одной стороны, необходимо обрабатывать большие объемы данных. С другой стороны, различные задачи обработки последовательностей ДНК сводятся к сравнению двух последовательностей нуклеотидов. В этой работе мы рассмотрим используемые представления последовательностей нуклеотидов и их кодировку для использования системой cuSTAR.

Ключевые слова: ассоциативные параллельные алгоритмы, биоинформатика, GPU, CUDA.

Введение. За последние несколько лет обработка генома стала широко востребованной задачей. Различными вариантами обработки занимаются как частные лаборатории (от выполнения ДНК-тестов до создания генетических паспортов), так и научные коллективы. При этом и первые, и вторые обрабатывают большие объемы данных как за счет количества образцов, так и за счет длины этих образцов: от десятков тысяч до нескольких миллиардов нуклеотидов. Кроме этого, расширяется и область применения молекулярно-генетических исследований (биомедицина, фармакология, нанобиоинженерия и т. д.). Заметим, что огромная часть вычислений связана с поиском отдельных нуклеотидов или их последовательностей как в большей последовательности, так и в большом числе последовательностей.

Для таких вычислений целесообразно использовать ассоциативные параллельные архитектуры, поскольку они специализируются на быстром поиске. Но такие системы слабо представлены на рынке компьютерной техники, в отличие от широкодоступных графических ускорителей. Библиотека cuSTAR была разработана для реализации абстрактной модели ассоциативных вычислений (STAR-машины) на графических ускорителях [1, 2].

Исследование выполнено в рамках государственного задания ИВМ и МГ СО РАН 0251-2021-0005. Работа была представлена на международной конференции «Марчуковские научные чтения–2022».

В данной работе предлагается несколько методов организации данных, позволяющих использовать ассоциативные алгоритмы для решения различных задач, связанных с обработкой генома.

В первом разделе мы напомним краткое описание ассоциативной модели STAR-машины и ее реализации cuSTAR. Во втором разделе предлагаются способы кодирования последовательностей нуклеотидов, которые минимизируют или объем требуемой памяти, или время поиска нуклеотидов в последовательности. В третьем разделе представлен способ построения графа де Брейна по исходной последовательности нуклеотидов.

1. Краткое описание библиотеки cuSTAR, реализующей абстрактную ассоциативную модель вычислений. Абстрактная ассоциативная модель вычислений STAR-машина, которая реализуется библиотекой cuSTAR, для ассоциативной обработки использует следующие типы данных: Table (бинарная таблица), Slice/слайс (битовый столбец) и word/слово (битовая строка). Над этими типами определены операции: чтение и запись столбцов и строк таблицы, побитовые логические операции и сдвиг для переменных типов Slice или Word. Кроме этого, для переменной X типа Slice определена операция $FND(X)$, возвращающая позицию старшей единицы в слайсе. Эта операция позволяет организовывать неравномерный цикл и широко используется в построении ассоциативных алгоритмов.

Библиотека cuSTAR позволяет выполнять ассоциативные алгоритмы на неассоциативной архитектуре (графических ускорителях), сохраняя ассоциативные преимущества. Фактически вычисления производятся параллельно над битовыми векторами. В определенных случаях переменная типа Table может быть обработана параллельно не только по столбцам, но и все столбцы обрабатываются также параллельно. К достоинствам системы cuSTAR относятся возможность задать произвольный размер векторов, работа на языке высокого уровня и разработанные рекомендации для оптимизации ассоциативных алгоритмов для исполнения на GPU.

Хотя cuSTAR, как и модель, которую она реализует, универсальна, для решения задач биоинформатики имеет смысл выделить в отдельные модули способы хранения данных и алгоритмы их обработки, учитывающие специфику области.

2. Два возможных представления нуклеотидной последовательности для ассоциативной обработки. Модель STAR-машины предполагает представление данных в табличном виде, при этом существенны следующие два факта: строки бинарны, обработка данных выполняется преимущественно с помощью битовых столбцов. Поэтому представление данных в системе cuSTAR принципиально другое. На рис. 1 показаны общепринятое представление последовательности нуклеотидов (строка типа char) и способ представления в системе cuSTAR (объект типа Table — двумерный массив элементов типа unsigned long long int).

В данной статье мы сосредоточимся только на представлении последовательности нуклеотидов.

2.1. Представление, оптимальное по занимаемой памяти. Алфавит нуклеотидов состоит из символов А (аденин), С (цитозин), Г (гуанин) и Т (тимин). Также часто в данных используется символ «—» для обозначения возможных пробелов в чтении, вставок или удалений в последовательности нуклеотидов. Таким образом используется четыре или пять символов в зависимости от поставленной задачи.

строка из 128 нуклеотидов		переменная типа Table			
строка	типа char	0	1	2	3
a	1100001	0	0	1	
t	1110100	1	1	1	
...	...	:	:	:	
c	1100011	0	1	1	
a	1100001	0	0	1	
t	1110100	1	1	1	
g	1100111	1	0	1	
...	...	:	:	:	
a	1100001	0	0	1	
t	1110100	1	1	1	
		64 элемента			
		48 байт			
		128 байт			

Рис. 1. Стандартное представление массива нуклеотидов и представление для ассоциативного алгоритма

Таблица 1

Кодирование нуклеотидов.

Оптимальный метод по
минимизации объема памяти

символы	—	A	C	G	T
бинарный код	000	001	011	101	111

Таблица 2

Кодирование нуклеотидов.

Оптимальный метод по
времени обработки данных

символы	—	A	C	G	T
бинарный код	0000	1000	0100	0010	0001

Поэтому будем ориентироваться на пятисимвольный алфавит. Заметим, что при кодировании алфавита приведенным в табл. 1 способом последний столбец состоит только из «1», если в данных не присутствует символ «—». В этом случае его можно не обрабатывать.

2.2. Представление, оптимальное для обработки данных. Многие задачи биоинформатики предполагают поиск позиций нуклеотидов в последовательности. Таким образом, происходит многократный поиск одних и тех же символов в последовательности. Но если оптимизировать представление данных не по объему предполагаемой памяти, а исходить от необходимости минимизировать время поиска, то предпочтительней другое представление нуклеотидов.

С одной стороны, данный способ (табл. 2) требует для записи нуклеотидов 4 бита вместо 3-х, что увеличивает объем памяти на 22 % на CPU и на 33 % на GPU. С другой стороны, каждому нуклеотиду соответствует определенный столбец таблицы, а позиции символа «—» легко определяются по формуле $NA = \neg(\vee_{k=1}^4 col(k, GEN))$, где GEN — таблица данных, а $col(k, GEN)$ — k -й столбец этой таблицы. И для определения всех позиций

Таблица 3

Сравнение представлений по объему занимаемой памяти

метод	CPU(B)	GPU(KB)					
		100	1 000	10 000	100 000	1 000 000	10 000 000
Mem_opt	72	0.05	0.38	3.68	36.63	366.21	3662.11
Time_opt	88	0.06	0.50	4.91	48.84	488.28	4882.81
char[]	8	0.10	0.98	9.77	97.66	976.56	9765.63

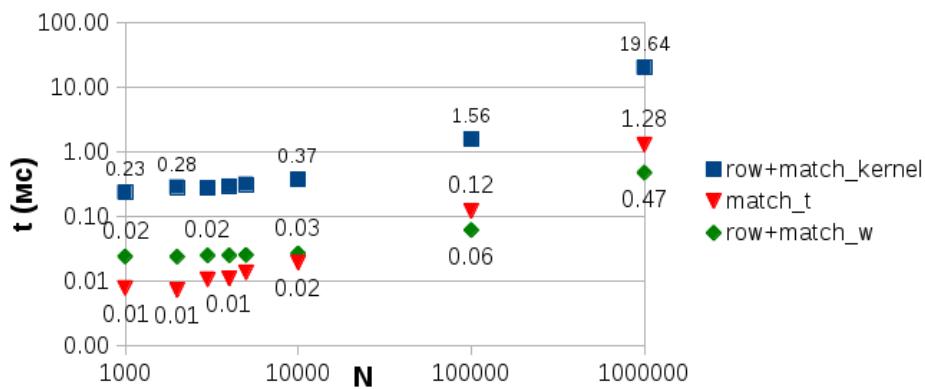


Рис. 2. Сравнение времени поиска нуклеотида в последовательности

заданного нуклеотида в таблице GEN вместо поиска слова в таблице требуется определить номер столбца, который ему соответствует, что существенно сокращает время этой операции. При этом стоит отметить, что стандартное представление последовательности нуклеотидов как символьный массив занимает в 2 раза больше памяти на GPU по сравнению с четырехбитовым представлением.

На рис. 2 показано время поиска нуклеотида в последовательности для обоих представлений. При использовании представления, оптимизированного по памяти, поиск выполняется процедурой $match_kernel$ из библиотеки стандартных процедур. При использовании представления, оптимизированного для обработки данных, поиск может быть выполнен двумя способами: $match_t$ — поиск позиций строк в таблице G , совпадающих с j — строкой в таблице P ; $match_w$ — поиск позиций строк в таблице G , совпадающих со словом $w = row(P, j)$. Отметим, что при выполнении $match_kernel$ и $match_w$ учитывалось время извлечения слова w из таблицы P . Расчеты проводились на карте GeForce 920M. Видно, что второе представление действительно дает возможность выполнять поиск нуклеотида существенно быстрее (\approx в 10 раз и более). Линейный рост времени от $N = 10000$ связан с нехваткой ресурсов видеокарты для выполнения одновременно всех ядер процедур. Тем не менее, поиск всех позиций нуклеотида в последовательности 1 млн выполняется за 19, 1,3 и 0,5 миллисекунд соответственно. Также необходимо отметить, что более современные графические ускорители имеют гораздо большие ресурсы как по памяти, так и по количеству ядер.

Тем не менее, оба представления полезны. Представление $time_opt$ предпочтительней для использования в тех алгоритмах, где производится непосредственное сравнение последовательностей нуклеотидов. Но в большом числе алгоритмов выполняется обработ-

		atcaatgatcaagcttctaaggatcg			
4-меры (дуги)				5-меры (дуги)	
		left	right	left	right
a	t	c	a	001 111 011	111 011 001
t	c	a	a	111 011 001	011 001 001
c	a	a	t	011 001 001	001 001 111
a	a	t	g	001 001 111	001 111 101
a	t	g	a	001 111 101	111 101 001
t	g	a	t	111 101 001	101 001 111
g	a	t	c	101 001 111	001 111 011
a	t	c	a	001 111 011	111 011 001
t	c	a	a	111 011 001	011 001 001
c	a	a	g	011 001 001	001 001 101
a	a	g	c	001 001 101	001 101 011
a	g	c	c	001 101 011	101 011 111
a	g	c	t	101 011 111	011 111 111
g	c	t	t	011 111 111	111 111 011
c	t	t	c	011 111 011	111 111 011
t	t	c	t	111 111 011	111 011 111
t	c	t	a	111 011 111	011 111 001
c	t	a	a	011 111 001	111 001 001
t	a	a	g	111 001 001	001 001 101
a	a	g	c	001 001 101	001 101 011
a	g	c	a	001 101 011	101 011 001
g	c	a	t	101 011 001	011 001 111
c	a	t	g	011 001 111	001 111 101

Рис. 3. Представление графа де Брейна для $k = 3$ и для $k = 4$

ка графов, построенных из первоначальных последовательностей. И в этом случае будет предпочтительней использование представления `mem_opt`.

3. Построение графа де Брейна. Как уже упоминалось, в некоторых задачах обрабатываются не сами последовательности нуклеотидов, а построенные по ним графовые структуры. Одна из таких структур — граф де Брейна. В частности, граф де Брейна широко используется в инструментах сборки последовательностей (сборщики генома Velvet, ALLPATHS, Abyss, JR-Assembler, EPGA и другие) [3].

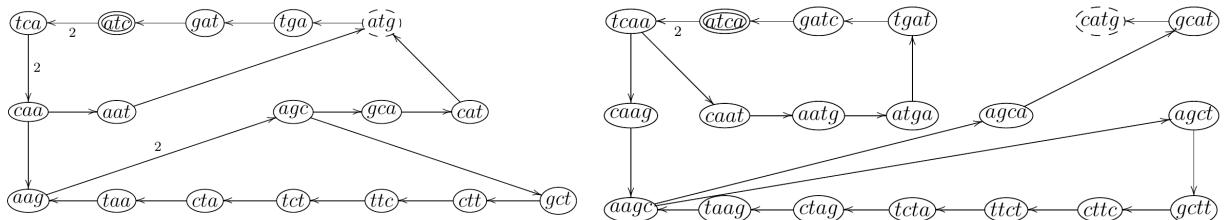
Определение 1. Графом де Брейна для натуральных параметров k и n называется ориентированный граф, вершинам которого соответствуют все возможные n -значные k -ичные последовательности, а ребра соединяют те и только те пары вершин, для которых последние $n-1$ цифр первого числа совпадают с первыми $n-1$ цифрами второго числа.

Отметим, что при обработке последовательностей нуклеотидов параметр n фиксируется как $n = k + 1$.

Определение 2. k -мер — последовательность из k символов. Вершинами графа де Брейна служат все k -меры, а дугами все $k + 1$ -меры. Например, при $k = 3$ 4-мер *atca* соответствует дуге *atc*—>*tca*.

При использовании cuSTAR построить граф де Брейна по последовательности нуклеотидов любого параметра k просто. Граф при этом задается списком ребер. Такой способ является наряду с матрицей весов (или матрицей смежности) стандартным представлением для ассоциативной обработки. Отметим, что, задавая граф в виде списка ребер, мы избегаем проблем, связанных с повтором дуг.

1) при чтении последовательности формируется таблица GEN размера $3 \times length$, где $length$ — длина последовательности.

Рис. 4. Графы де Брейна для $k = 3$ и для $k = 4$

2) для графа, заданного списком дуг, формируем таблицы $left$ и $right$ размера $3k \times (length - k)$:

- таблица $left$ получается копированием столбцов матрицы GEN в соответствующие столбцы со сдвигом вверх.
- в свою очередь, таблица $right$ получается копированием со сдвигом вверх на одну строку таблицы $left$.

На рис. 3 представлены представления двух графов де Брейна одной и той же последовательности нуклеотидов для $k = 3$ и $k = 4$ соответственно. На рис. 4 представлены сами графы для этой последовательности. И тут необходимо отметить следующие моменты.

С одной стороны, чем больше значение k , тем длиннее бинарный код для вершин ($3k$). Это влечет за собой увеличение времени поиска вершины при обработке графа, поскольку алгоритм поиска строки в таблице зависит от ее ширины.

Но, с другой стороны, структура самого графа при этом упрощается. В нем образуется меньше циклов, что будет уменьшать общее время обработки. Так при восстановлении нуклеотидной последовательности по графикам де Брейна (поиск Эйлерова пути) для $k = 3$ будут восстановлены следующие возможные варианты:

$$atcaatgatcaagttctaaagcatg, atcaagttctaaagcatgatcaatg.$$

Для $k = 4$ последовательность восстанавливается однозначно

$$atcaatgatcaagttctaaagcatg.$$

Отметим, что особенность решения задач биоинформатики в том, что вычисления призваны не найти точное решение, а сократить количество возможных вариантов, которые потом проверяются биологами. При этом при решении реальных задач количество возможных вариантов может доходить до сотен и тысяч.

Заключение. Данная работа представляет возможные способы организации данных биоинформатики для обработки системой сиSTAR. Это первая работа из предполагаемого цикла, посвященного разработке ассоциативных алгоритмов для решения задач биоинформатики.

Предложенные способы кодирования данных компактнее стандартного представления в виде массива символов. Кодирование, оптимальное для обработки, позволяет производить поиск нуклеотидов в последовательности на порядок быстрее, чем процедура из библиотеки стандартных ассоциативных алгоритмов. Кодирование, оптимальное по занимаемой памяти, предпочтительней в случае, если используется представление нуклеотидной последовательности в виде графа.

Так как обработка генома предполагает множественный поиск по большому объему данных, разработка ассоциативных алгоритмов для этой области актуальна. Прикладная

ценность работы состоит в возможности выполнения данных алгоритмов на графических ускорителях — широко распространенном оборудовании от персональных компьютеров до кластерных систем.

Список литературы

1. Снытникова Т. В., Непомнящая А. Ш. Решение задач на графах с помощью star-машины, реализуемой на графических ускорителях // Прикладная дискретная математика. 2016. № 3(33). С. 98–115.
2. Снытникова Т. В. Реализация модели ассоциативных вычислений на GPU: библиотека базовых процедур языка star // Вычислительные методы и программирование. Новые вычислительные технологии. 2018. № 19. С. 85–95.
3. COMPEAU Ph., PEVZNER P. A., TESLER G. How to apply de bruijn graphs to genome assembly // Nature Biotechnology. 2011. V. 29(11). P. 987–991.



Снытникова Татьяна Валентиновна — младший научный сотрудник лаборатории синтеза параллельных программ Института вычислительной математики и математической геофизики. Область профессиональных интересов: ассоциативные параллельные алгоритмы, реа-

лизация ассоциативных алгоритмов на графических ускорителях.

Snytnikova Tatiana Valentinovna is a junior researcher at the Laboratory of Synthesis of Parallel Programs of the Institute of Computational Mathematics and Mathematical Geophysics. The field of professional interests is associated with associative parallel algorithms, the implementation of associative algorithms on graphics accelerators.

Дата поступления — 16.11.2022