Нижегородский государственный университет им. Н.И. Лобачевского

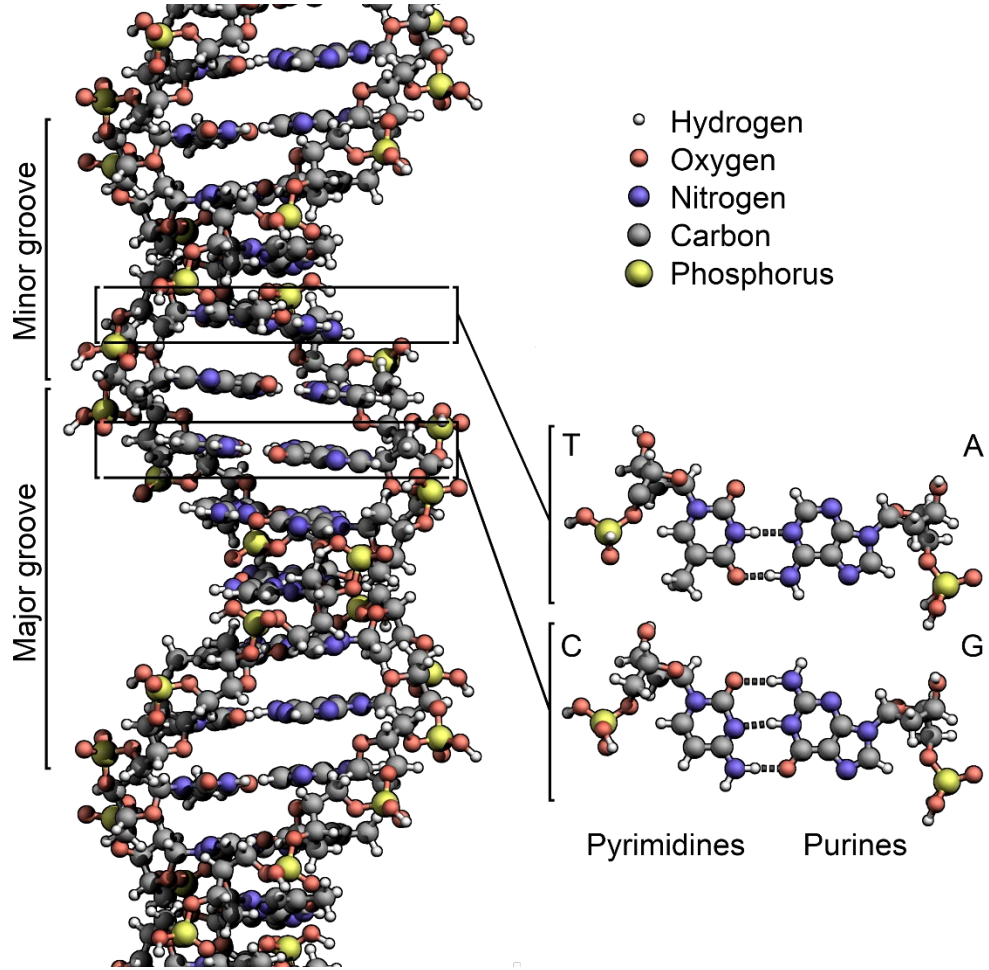
Разработка бенчмарка для устройств на архитектуре RISC-V на основе одной задачи биоинформатики

М.А. Козлов, В.Д. Волокитин,
Е.А. Панова, И.Б. Мееров

RuSCDays'2024
23.09.2024

Предметная область

- Предметная область – биоинформатика
- ДНК состоит из азотистых оснований: аденина, гуанина, цитозина и тимина
- Часто встречающиеся подпоследовательности ДНК небольшой длины могут нести в себе важную генетическую информацию
- Задача – исследование ДНК на наличие таких подпоследовательностей
- Предмет доклада – разработка бенчмарка для CPU на архитектуре RISC-V на основе данной задачи



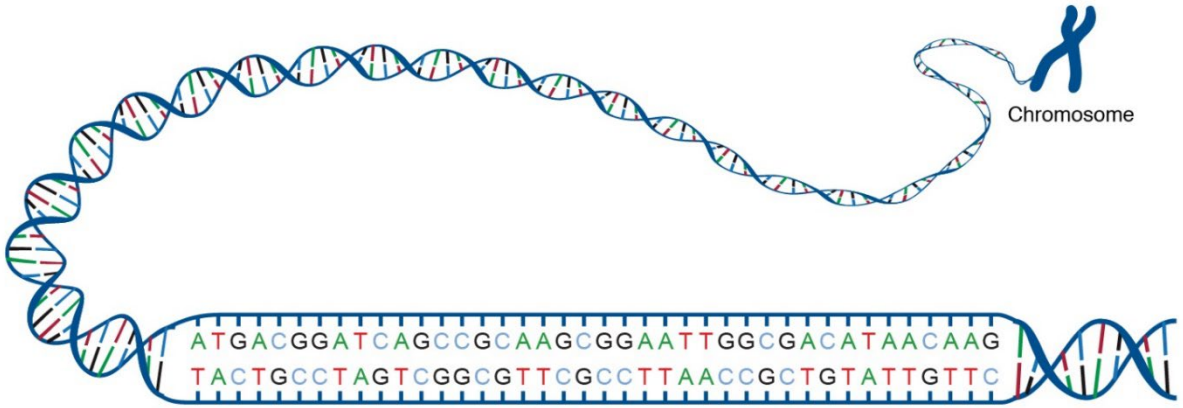
Работа поддержана Минобрнауки РФ, проект № FSWR-2023-0034. Авторы благодарят ННГУ за доступ к вычислительным ресурсам суперкомпьютера «Лобачевский».

Сведение к алгоритмической задаче

- Каждому азотистому основанию можно сопоставить букву латинского алфавита (“A”, “C”, “G”, “T”)
- С учетом представления азотистых оснований в виде символов, исходная задача сводится к задаче поиска наиболее часто встречающихся подстрок фиксированной длины в строке

Особенности задачи

- Отсутствие операций с числами с плавающей точкой
- Большое количество операций по работе с памятью



Исследуемые алгоритмы решения

Наивный алгоритм

- Наибольшее количество посимвольных сравнений строк

Алгоритм Рабина-Карпа

- Использует хэширование
- В качестве хэш-функции были выбраны два варианта:
  - Кольцевой полиномиальный хэш
  - Функция SWAR, сравнивающая два первых и последних символа

Hash3

- Хэширует первые 3 символа искомого паттерна, в хэш-таблицу записывает перемещения до конца паттерна
- Хорошо работает на маленьких алфавитах

Детали реализации

- Мы разработали *скалярную* и *векторную* версии каждого алгоритма для CPU на архитектуре RISC-V.
- Для сравнения производительности аналогичные реализации были разработаны для архитектуры x86.
- Во всех алгоритмах производится посимвольное сравнение строк в векторном режиме. В алгоритме Рабина-Карпа с хэш-функцией SWAR выполняется векторное сравнение двух первых и последних символов.
- Для векторизации на процессоре RISC-V использовались векторные *интринсики*, компилирующийся в инструкции RVV-0.7.1

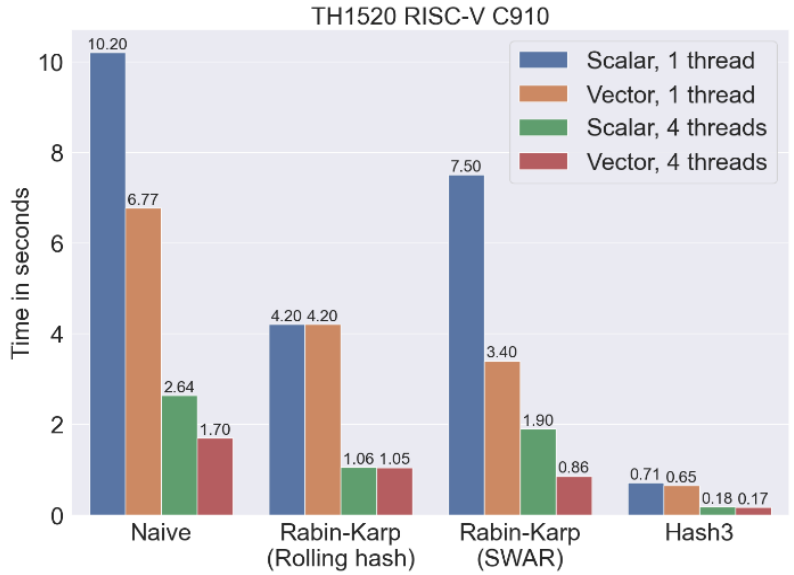
Оптимизации, связанные с предметной областью

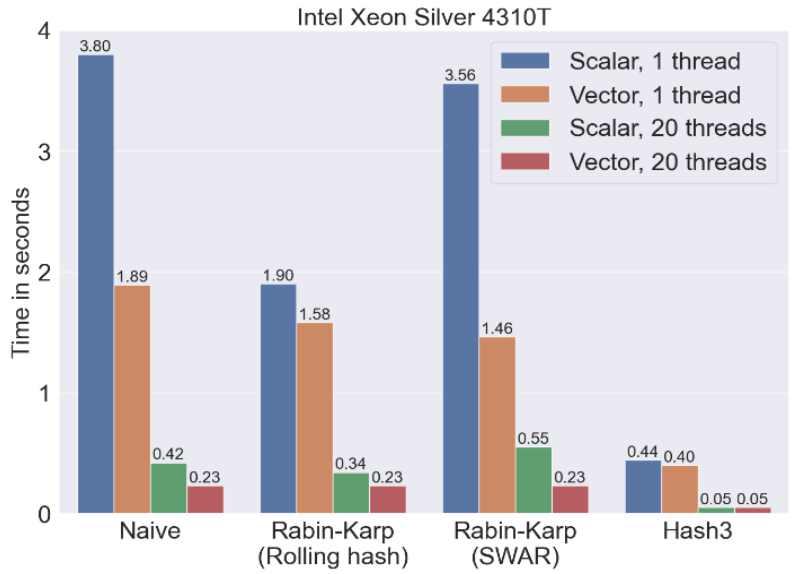
- В алгоритме Hash3 рассматриваются подстроки длины 3, которых в четырехбуквенном алфавите всего 256. Следовательно, мы использовали
  хэш-таблицу размером 256, которая целиком убирается в кэш
- В оригинальной постановке функция SWAR сравнивает только первый и последний символы, но и-за маленькой мощности алфавита такая хэш-функция предотвращает совсем не большое число посимвольных сравнений подстрок. Поэтому мы сравниваем 2 первых и последних символа.

Результаты экспериментов

Результаты

- На RISC-V как у скалярных, так и у векторных реализаций наблюдается *линейное ускорение* алгоритмов при распараллеливании
- Наибольшее ускорение от векторизации получили наивный алгоритм и алгоритм Рабина-Карпа с хэш-функцией SWAR, в 1.5 и 2.2 раза соответственно
- Результаты ускорения строковых алгоритмов на RISC-V сравнимы с результатами, полученными на x86. Наибольшее ускорение от векторизации на x86 составило 2,4 раза, на RISC-V - 2,2 раза. CPU на RISC-V получил линейное ускорение от параллелизма на 4 потоках, в то время как на x86 алгоритмы на 20 потоках стали работать быстрее в 5,6-9 раз.





- *Тестовая конфигурация x86 (Intel Xeon Silver 4310T, 2.30ГГц, 2x10 ядер, AVX512, 64 ГБ DDR4-2667).*
- *Тестовая конфигурация RISC-V (TH1520, 2.0ГГц, 4 ядра Xuantie C910, 16 ГБ LPDDR4X-3733, RVV-0.7.1)*
- *Размер исходного фрагмента ДНК 43794, размер искомого паттерна 128.*

Литература

1. Козлов М. А., Панова Е. А., Мееров И. Б. Реализация поиска наиболее часто встречающихся последовательностей ДНК с использованием библиотеки Kokkos. //Проблемы информатики. – 2024.
2. Plaxton G. String Matching: Rabin-Karp Algorithm //Theory in Programming Practice. University of Austin, Texas. – 2005.
3. Kuznetsov A. V., Maysnikov V. V. A fast plain copy-move detection algorithm based on structural pattern and 2D Rabin-Karp rolling hash //Image Analysis and Recognition: ICIAR 2014, Proceedings, Part I 11. – Springer International Publishing, 2014. – P. 461-468.
4. Mula W. SIMD-friendly algorithms for substring searching. URL: http://0x80.pl/articles/simd-strfind.html (дата обращения: 25.03.2024).
5. Lecroq T. Fast exact string matching algorithms //Information Processing Letters. – 2007. – Т. 102. – №. 6. – С. 229-235.
6. Репозиторий проекта. URL: https://github.com/Mishaizlesa/most_common_string (дата обращения: 25.03.2024).