Санкт-Петербургский государственный университет
Кафедра системного программирования

Оптимизация библиотеки xxHash для архитектуры RISC-V

Николай Алексеевич Пономарев, группа 21.Б10-мм

Научный руководитель: К. К. Смирнов, ст. преподаватель кафедры ИАС

Санкт-Петербург
2023

xxHash

xxHash — современная библиотека для высокопроизводительного хеширования
Особенности:

- Поддерживает вычисление 32-, 64- и 128-битных хешей
- Скорость работы превышает пропускную способность оперативной памяти[1]
- Имеет поддержку векторных расширений процессора
  - SSE2, AVX2, AVX512 (x86_64)
  - NEON, SVE (ARM)
  - VSX (PowerPC)
  - Но нет поддержки RVV (RISC-V)

[1] https://github.com/Cyan4973/xxHash/blob/v0.8.1/README.md

Постановка задачи

Целью работы является реализация хеш-функций из библиотеки xxHash при помощи векторных расширений архитектуры RISC-V.

Задачи:

- Сравнить возможности разных версий векторного расширения RISC-V
- Выбрать целевую платформу для адаптации кода и проведения измерений
- Адаптировать одну из существующих реализаций под выбранную платформу
- Выполнить замеры производительности адаптированного кода

Векторные возможности RISC-V

Векторное расширение RISC-V, сокращенно RVV, в данный момент представлено двумя несовместимыми версиями:

RVV 1.0	RVV 0.7.1
Стабильная версия	Нестабильная версия
Поддерживается современными компиляторами	Требуется специализированный компилятор
Есть перегрузка intrinsic функций	Нет перегрузки intrinsic функций
Нет устройств в продаже[2]	Есть устройства в свободной продаже

[2] На момент написания

Целевая платформа

В качестве целевой платформы был выбран одноплатный ПК Sipeed Lichee RV на чипе Allwinner D1

- Единственная доступная на момент написания платформа с поддержкой RVV
- Поддерживает RVV 0.7.1
- Не поддерживает 64-битные элементы вектора $\Longrightarrow$ используем 32-битные

Проблемы при адаптации

В качестве базовой была выбрана реализация для SSE2, она использует векторы с 64-битными элементами. Адаптированная реализация использует 32-битные элементы вектора.

При реализации были встречены следующие проблемы:

- Сложности обработки 64-битных чисел, представленных как пары 32-битных
  - Необходимость реализации умножения с ручным расширением
  - Необходимость ручной реализации сложения с переносом
- Сложности с работой с масками в RVV 0.7.1
  - Отсутствие операции загрузки маски из памяти, необходимость создания маски по уже загруженному вектору
- Работа с неопределенным поведением при загрузке по невыровненному адресу

Экспериментальное исследование

Измерения проводились на одноплатном компьютере Sipeed Lichee RV 86 со следующими характеристиками:

- Процессор Allwinner D1 с частотой 1 ГГц;
- Оперативная память DDR3 объемом 512 Мб с частотой 800 МГц;
- Операционная система Debian Sid с последними обновлениями на момент тестирования.

Для компиляции использовался компилятор от компании Alibaba с флагами `-O3` и `-march=rv64gcv0p7`. Для выбора набора функций использовались флаги `-DXXH_VECTOR=XXH_SCALAR` и `-DXXH_VECTOR=XXH_RVV` соответственно. Данные измерений были получены с помощью поставляемой вместе с библиотекой утилиты `xxhsum`.

Результаты экспериментального исследования

Таблица: Сравнение производительности хеш-функции XXH3 на входных данных размером в 1000 Кб; числа приведены с относительной погрешностью 0.5%

Набор функций	Скорость работы, Мб/с
Скалярный	169.3
Векторный	116.0

Вероятные причины замедления:

- Большое количество операций загрузки при вызове функций
- Использование инструкций объединения векторов и инструкций перестановки элементов вектора

Результаты

- Изучены возможности разных версий векторного расширения RISC-V
- Выбрана целевая платформа для адаптации кода и проведения измерений
- Проведена адаптация одной из существующих реализаций под выбранную платформу
- Выполнены замеры производительности адаптированного кода

Исходный код расположен по адресу: `https://github.com/WoWaster/xxHash`.