Санкт-Петербургский государственный университет

Кафедра системного программирования

Динамическая бинарная трансляция в RISC-V с помощью Instrew

Михайлов Илья Игоревич

Научный руководитель: к.ф.-м.н. Д. В. Луцив, доцент кафедры системного программирования
Консультант: В. А. Кутуев, инженер-исследователь, Лаборатория технологий программирования инфраструктурных решений СПбГУ

Рецензент: Ф. В. Сергеев, исполнительный директор ПАО «Сбербанк России»

Санкт-Петербург
2025

Введение

- Бинарные трансляторы используются для эмуляции, профиляции, анализа кода
- Важный класс — динамические бинарные трансляторы (далее — ДБТ)
  - Используются для эмуляции «на лету»
  - Например Rosetta 2
- Перспективно использовать LLVM IR в качестве промежуточного представления для ДБТ
  - Фреймворк LLVM позволяет проводить качественные оптимизации
- RISC-V — модульная открытая архитектура, где наличие производительного эмулятора поможет развитию экосистемы
- Предлагается портировать ДБТ с поднятием в LLVM IR на RISC-V

Обзор

- Динамические бинарные трансляторы:
  - QEMU-user
  - Box64
  - Valgrind
  - ...
- Динамические бинарные трансляторы с поднятием в LLVM IR:
  - Instrew + Rellume
  - BinRec, HQEMU
    - Не обновлялись с 2022 и 2018 года соответственно
  - DBILL
    - Код не был найден

Выводы из обзора

Выбрали ДБТ Instrew для портирования, так как

- Из всех рассмотренных ДБТ с поднятием в LLVM IR, Instrew + Rellume единственные поддерживаются мейнтейнером
- Instrew производительнее, чем QEMU и Valgrind (основываясь на бенчмарках[1])
- Instrew имеет guest-поддержку RISC-V и AArch64, в отличие от Box64
- Instrew более расширяемый чем Box64
- Является open source проектом

[1] https://mediatum.ub.tum.de/doc/1614897/1614897.pdf

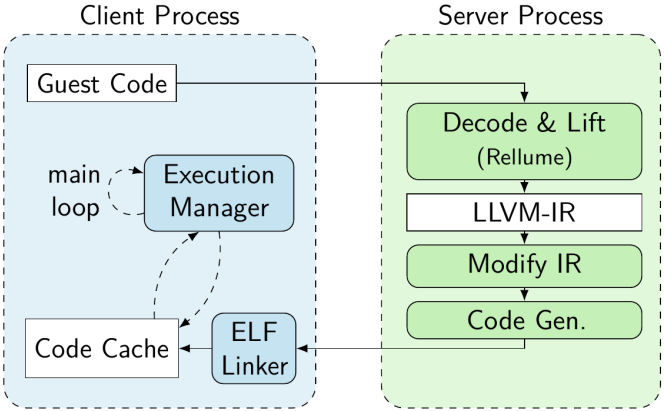
Постановка задачи

Цель: Добавить поддержку host-архитектуры RISC-V для ДБТ Instrew

Задачи:

1. Провести обзор архитектуры Instrew
2. Портировать Instrew на RISC-V
   - Исправить ошибки компиляции
   - Исправить ошибки, выявленные при внутреннем интеграционном тестировании в
     - Instrew
     - Rellume
3. Выполнить тестирование на Coreutils и исправить ошибки, выявленные при тестировании
4. Добавить в CI поддержку сборки и тестирования на RISC-V
5. Выполнить измерения производительности Instrew на xxHash и LINPACK и сравнить с QEMU-user и Box64

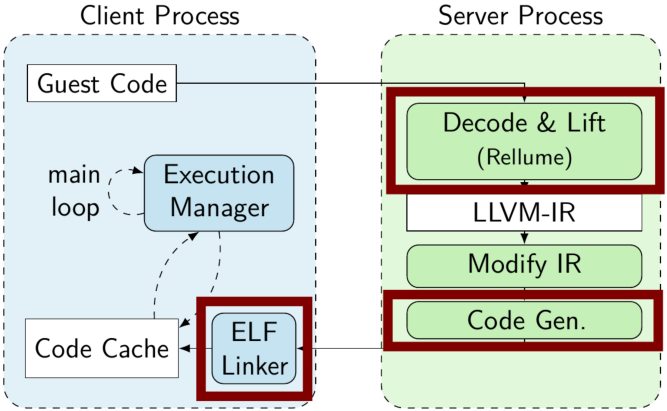
Архитектура Instrew



[2]

[2] https://mediatum.ub.tum.de/doc/1614897/1614897.pdf

Архитектура Instrew



[3]

[3] https://mediatum.ub.tum.de/doc/1614897/1614897.pdf

Архитектура Instrew: динамическая линковка

- Получение порожденного сервером ELF (Executable and Linkable Format)
- Линковка ELF-файла
  - Разрешение недостающих символов из PLT (Procedure Linkage Table)
  - Добавление PLT-трамплинов для символов из PLT
  - Разрешение релокаций — процесс вычисления и подстановки абсолютных адресов вместо относительных адресов из ELF
  - Копирование функции в адресное пространство, которое поддерживает исполнение кода
- Выполнение функции
  - Эмуляцией необходимых системных вызовов с помощью PLT-трамплинов

Портирование Instrew: компиляция

- Добавить архитектурно-зависимые участки кода
  - Точка входа на ассемблере
  - PLT-трамплины
  - Очистку кэша (Cache flush)
- Реализовать эмуляцию системных вызовов RISC-V

Портирование Instrew: внутренние тесты Instrew

- Изначально, ни один unit-тест не проходил
- Для исправления ошибок при тестировании, были добавлены
  - Зависимость libatomic, 3 вхождения в PLT и реализация 4-ёх типов релокаций
- Все тесты проходят успешно после исправления ошибок
- Сложно отлаживать релокации:
  - Профиляция памяти во время трансляции с помощью GDB

Портирование Instrew: внутренние тесты Rellume

- Некоторые внутренние тест-кейсы используют динамический линкер LLVM, которые завершаются с SEGFAULT на RISC-V
  - В следствие чего снижается покрытие тестами
- Обновили версию LLVM до 20, что потенциально может решить проблему
- Результат
  - Тесты все также не проходят
  - Но знаем номера недостающих релокаций в динамическом линкере LLVM

Тестирование Instrew

- Используем Coreutils в качестве набора тестовых программ
  - Внутренние тесты Coreutils
  - Выборочное тестирование конкретных утилит
- Для исправления ошибок было сделано:
  - Добавлено 14 вхождений в PLT
  - Добавлена реализация 8-ми типов релокаций
  - Добавлена эмуляция 13-ти системных вызовов x86_64
  - Добавлена эмуляция 10-ти системных вызовов riscv64

CI

- Есть трудности в связи с отсутствием нативной поддержки RISC-V в GH Actions
- Есть два варианта решения этой проблемы:
  - С помощью self-hosted runner на RISC-V
  - С помощью эмуляции RISC-V (например с помощью QEMU-user)
- Удалось опробовать оба решения, но был выбран второй подход, так как:
  - Self-hosted runner неудобен для мейнтейнера
  - Actions runner использует .NET, который на данный момент плохо поддерживает RISC-V
    - Не удалось запустить на виртуальной машине, только на плате

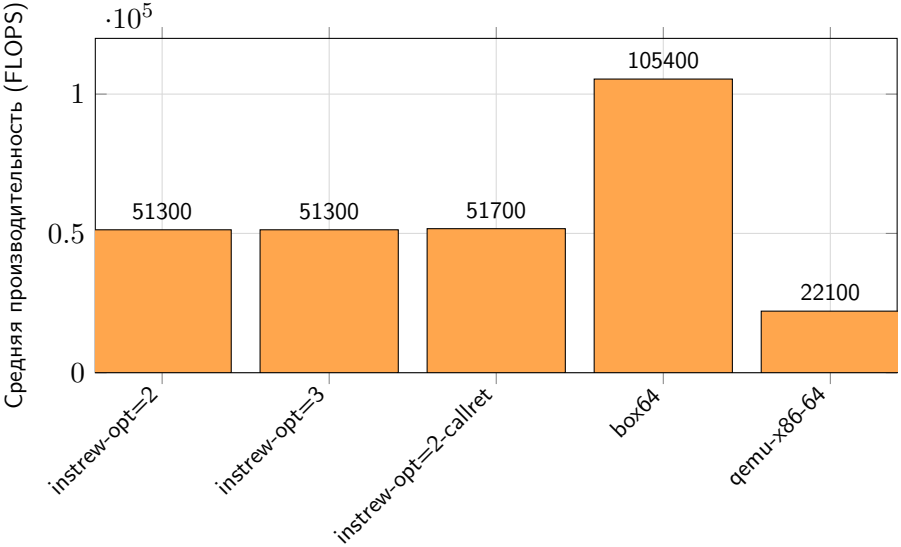
Замеры производительности

В оригинальной статье используется платный бенчмарк SPEC CPU 2017

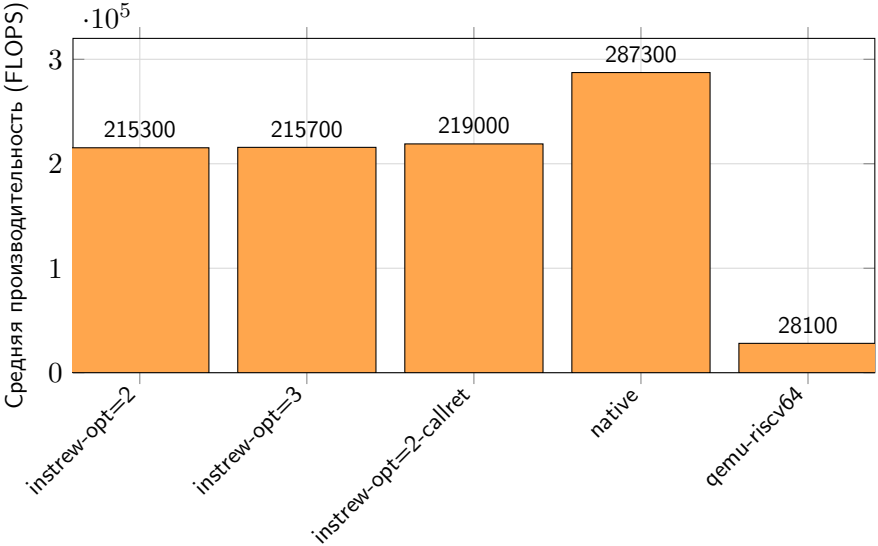
- Выбрали xxHash и LINPACK для замеров производительности
- LINPACK является стандартом для замеров производительности для линейной алгебры

Замеры производились на плате Banana Pi BPI-F3, процессор SpacemiT K1, 8 ядер, 4гб ОЗУ

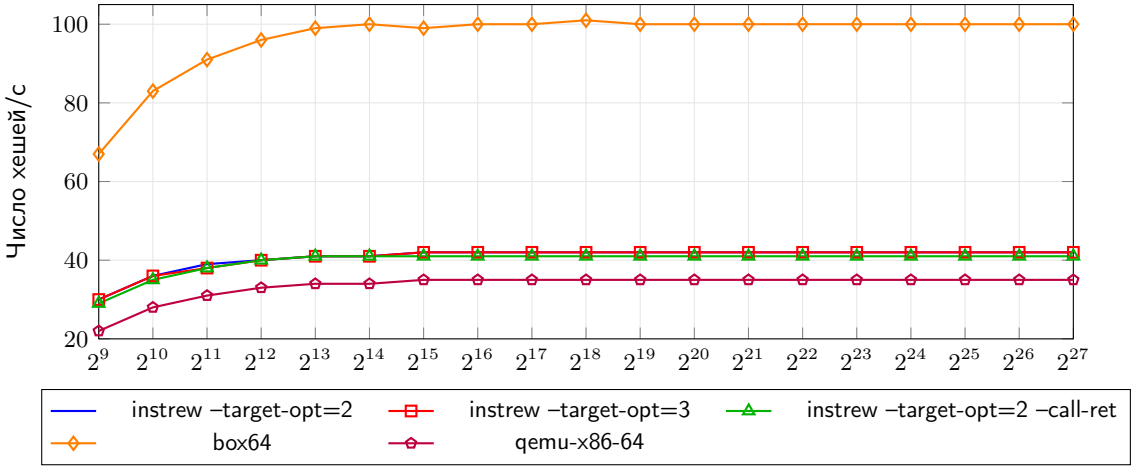
Сравнение на LINPACK: linpack для x86_64



Сравнение на LINPACK: linpack для riscv64



Сравнение на xxHash: benchHash для x86_64



Заключение

В результате работы были выполнены следующие задачи:

1. Выполнен обзор архитектуры Instrew
2. Instrew портирован на RISC-V, как хост архитектуру
3. Instrew протестирован на Coreutils
4. В CI добавлена поддержка сборки и тестирования на RISC-V и сделан пулл-реквест в основную ветку Instrew
5. Выполнены замеры производительности Instrew на xxHash, LINPACK и проведено сравнение с QEMU-user и Box64

Код доступен по данной ссылке: https://github.com/aengelke/instrew/pull/20.

Портирование Instrew: отладка

- Нестандартные флаги сборки, которые чувствительны к версии компилятора
  - Возникает конфликт между клиентом и динамическим линкером ld-linux.so
  - Так как клиент сам является динамическим линкером
- Нетривиальный способ запуска клиента
  - Загрузка процесса происходит через `fexecve` по умолчанию
  - Не понятно как отлаживать с помощью GDB
- Некоторые инструменты отладки не работают
  - Нельзя подключить ASan; в клиенте своя работа с памятью, не использующая интерфейс `malloc`
  - Valgrind имеет поддержку только стандартного расширения RISC-V (rv64gc)

Артефакты

Репозиторий с артефактами

- https://github.com/mikhaylovilya/instrew-artifacts

Результаты тестирования на Coreutils

Guest Архитектура	Статический cp	Статический cat	Статический factor
rv64	проходят 2 из 10 тестов	проходят 3 из 4 тестов	проходят 3 из 4 тестов
x86_64	проходят 5 из 10 тестов	проходят 3 из 4 тестов	проходят 3 из 4 тестов