Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра системного программирования
Программная инженерия

Влаев Никита Владиславович

Измерение и анализ времени запуска Unikernel операционных систем

Курсовая работа

Научный руководитель:
ассистент Козлов А.П.

Санкт-Петербург
2020

Оглавление

Введение

Unikernel [14] – это специализированные образы машин с единым адресным пространством, созданные с использованием библиотечных операционных систем. Unikernel'ы помогают упростить функциональность и снизить использование ресурсов облачных сервисов до необходимого минимума. Они создаются путем формирования специализированных машинных образов, которые запускаются на гипервизоре, таком как Xen [15], или прямо на устройствах. Поскольку гипервизоры обеспечивают работу большинства общедоступных облачных вычислительных инфраструктур, таких как Amazon EC2 [1], Unikernel'ы позволяют службам работать дешевле, безопаснее и иметь лучший контроль, чем с полным программным стеком.

Unikernel'ы предоставляют много преимуществ по сравнению с традиционной ОС, включая улучшенную безопасность, меньшие размеры, большую оптимизацию и более быстрое время загрузки, однако появляются также и недостатки, такие как необходимость обновления драйверов и сложность обеспечения корректного взаимодействия нескольких приложений, работающих одновременно.

Embox [5] представляет из себя модульную конфигурируемую ОС реального времени для встроенных систем, в которых особенно важны вышеперечисленные свойства, по этой причине они уже частично присущи Embox. Разработка проекта Embox Unikernel, в свою очередь, мотивирована тем, что в современных Unikernel'ах не хватает таких положительных характеристик Embox, как POSIX-совместимость и конфигурируемость. Для того, чтобы оценить производительность текущей версии Embox Unikernel по сравнению с другими Unikernel'ами, обладающими POSIX-совместимостью, и оптимизировать составляющие, работающие недостаточно эффективно, необходимо произвести определение сравнительных характеристик производительности операционных систем, их подсчет и анализ. При этом, после оптимизации ключевые свойства Unikernel, описанные выше, не должны ухудшаться. Так как время запуска ОС до старта выполнения полезного приложения являет-

ся одной из важнейших и часто используемых характеристик unikernel, она является объектом исследования этой работы.

1. Постановка задачи

Целью данной работы является установка полезной функции, для которой быстрый запуск ОС важен, оценка производительности Embox путем анализа времени запуска функции в сравнении с другими Unikernel ОС и оптимизация компонентов Embox, используемых в этой функции.

Для достижения данной цели в рамках работы были поставлены следующие задачи:

1. Выбор Unikernel'ов
2. Проанализировать подходы к определению характеристики времени запуска:
   - Выбор полезного приложения
   - Выбор методики сравнения
     (a) Выбор среды
     (b) Выбор средства измерения времени
3. Измерить время запуска Embox и других субъектов
4. Проанализировать результаты
5. Оптимизировать компоненты Embox

2. Обзор

2.1. Выбор Unikernel'ов

2.1.1. ClickOS[2]

Высокопроизводительная операционная система с открытым исходным кодом, предназначенная для виртуализации устройств, инспектирующих и модифицирующих сетевой трафик. Функциональность полностью задается конфигурацией, не POSIX-совместима.

2.1.2. CliveOS[3]

Открытая операционная система, нацеленная на работу в распределённых и облачных системах. Системные интерфейсы разработаны в стиле CSPссылка. Приложения и компоненты взаимодействуют по каналам, а каналы соединяются с сетью, конвейерами и любыми другими артефактами ввода-вывода. Нет нативного ядра, код написан на Go, может исполнять программы на Go. Не POSIX-совместима.

2.1.3. Drawbridge[4]

Исследовательский прототип для виртуализации "песочницы" для приложений. Drawbridge сочетает в себе две основные технологии: picoprocess, который представляет собой изолированный контейнер на основе процессов с минимальной поверхностью API ядра, и ОС-библиотеку, которая является версией Windows, созданной для эффективной работы в рамках процесса picoprocess. Нет открытого доступа. Не POSIX-совместима.

2.1.4. HaLVM[6]

Порт набора инструментов компилятора Glasgow Haskell, который позволяет разработчикам создавать легковесные виртуальные машины, которые могут работать непосредственно на гипервизоре Xen. Программы пишутся на Haskell. Не POSIX-совместима.

2.1.5. IncludeOS[8]

Минималистичная, ориентированная на сервисные архитектуры, библиотечная операционная система для облачных сервисов. В настоящее время это исследовательский проект для запуска программ на C/C++ на виртуальных машинах. Неполная совместимость с Linux с помощью библиотеки Muslссылка. Не POSIX-совместима.

2.1.6. MirageOS[10]

Библиотечная операционная система, которая создает unikernel'ы для безопасных высокопроизводительных сетевых приложений для облачных вычислений и мобильных платформ. Код может быть разработан на обычной ОС, такой как Linux или MacOS X, а затем скомпилирован в полностью автономный специализированный unikernel, который работает под гипервизором Xen или KVM. MirageOS использует язык OCaml вместе с библиотеками, которые обеспечивают поддержку сетей, различных хранилищ и параллелизма. Не POSIX-совместима.

2.1.7. OSv[11]

Универсальный модульный unikernel, предназначенный для безопасного запуска немодифицированных приложений Linux на небольших виртуальных машинах в облаке. Создан с нуля для легкого развертывания и управления микро-сервисами и безсерверными приложениями с высокой производительностью. OSv поддерживает множество языков, включая немодифицированный JVM-bytecode, Python 2 и 3, Node.JS, Ruby, Erlang, а также языки, компилируемые непосредственно в машинный код, такой как Golang и Rust. Частично POSIX-совместима, однако нет прямой конфигурируемости.

2.1.8. Rumprun[13]

Программный стек, который позволяет запускать существующее неизмененное программное обеспечение POSIX в качестве unikernel. Rumprun

поддерживает несколько платформ, включая аппаратное обеспечение и гипервизоры, такие как Xen и KVM. Он основан на rumpkernel, который предоставляет конфигурируемый набор из драйверов из ядра NetBSD, таких как драйверы файловых систем, обработчики системных вызовов POSIX, драйверы устройств PCI, стек протоколов SCSI, virtio и стек TCP/IP. POSIX-совместим.

2.1.9. runtime.js[17]

Библиотечная операционная система для облака с открытым исходным кодом, работающая на JavaScript. Она может быть связана с приложением и развернута как легковесный образ виртуальной машины. Он построен на движке V8 JavaScript и использует управляемую событиями и неблокирующую модель ввода/вывода, основанную на Node.js. На данный момент KVM является единственным поддерживаемым гипервизором. Не POSIX-совместима.

2.1.10. HermitCore[7]

Исследовательский проект - новая операционная система Unikernel, предназначенная для масштабируемого и предсказуемого поведения в среде HPC и облачных средах. Текущая версия поддерживает C/C++, Fortran, Go, Pthreads, OpenMP и iRCCE в качестве библиотеки для передачи сообщений. Не POSIX-совместим.

2.1.11. Итог

Мотивацией выбора субъектов для сравнения в этой работе послужит схожесть по структуре и выполняемым задачам с Embox, чтобы исследование не свелось к сравнению «яблок и апельсинов». Например, ClickOS [2] или IncludeOS [8] предсказуемо выиграют сравнение как минимум по времени запуска, так как их разработка была направлена на минимизирование этого параметра. Поэтому есть смысл делать бенчмарк с Unikernel'ом, в вышеупомянутом смысле похожем на Embox.

В качестве такого варианта идеально подходит проект Rumpkernel в силу своего упора на модульность, конфигурируемость и POSIX-совместимость. В дальнейшем исследовании будут рассмотрены другие субъекты.

2.2. Выбор полезного приложения

В качестве полезного приложения был выбран httpd - легковесный http-сервер. HTTP - самый популярный протокол обмена данными в интернете, поэтому функция http-сервера действительно полезная, и для сервера время запуска ОС это критичный параметр.

Была выбрана именно эта реализация в силу ее встроенности в Embox и хорошо задокументированного процесса портирования POSIX-приложений под Rumpkernel. Кроме этого, в силу ограниченной функциональности этого сервера легко проверить корректность его работы.

2.3. Выбор методики сравнения

2.3.1. Выбор среды

Самые частые варианты окружения для запуска Unikernel – это гипервизоры, реже реальные устройства.

Отталкиваясь от выбора Unikernel'ов для сравнения, можно пересечь множества платформ которые поддерживают Embox и Rumpkernel, и в этом пересечении останется запуск под гипервизором Xen [15] и QEMU [12].

Так как в период начала и в течении этой работы в Embox на Xen не было в полной мере реализовано сетевое взаимодействие, невозможно было бы удостовериться в корректном поведении запущенного приложения, поэтому было решено выбрать QEMU.

Однако, для QEMU на момент исследования не существует специализированных инструментов для профилирования, поэтому было принятно решение исследовать Embox в процессе его оптимизации также

и на другой платформе, используя переносимость некоторых результатов.

В качестве кандидатов для такой платформы были выбраны Linux [9] и Xen по причине надежности и развитого сообщества, которое позволяет более быстро и качественно провести исследование.

В качестве критерия для выбора подойдет количество инструментов для анализа производительности приложения. На первый взгляд, такой способ выбора может показаться недостаточным, но количество инструментов на самом деле характеризует популярность платформы в сообществе, а значит готовность создавать и улучшать эти инструменты, и возможность найти инструмент или набор инструментов, необходимый и достаточный для решения специфичной задачи, которая может возникнуть в ходе исследования на этапе анализа и оптимизации.

Имеет смысл подсчета не всех инструментов, а тех, у которых есть признаки активного использования сообществом. Например, issues в репозитории (в случае open-source разработки), вопросы на stackoverflow.com и других форумах, и советы для использования реальными людьми.

	Профиляторы	Трейсеры	Дебаггеры
Xen	Intel VTune Xenoprof oprofile opcontrol uniprof	Xentrace Xenanalyze	Xendbg wenzel
Linux	perf valgrind strace ltrace callgrind oprofile	kprobes uprobes lttng ftrace	gdb ddd nemiver valgrind

Таблица 1: Сравнение количества инструментов для анализа процесса под Linux и в Xen

В итоге, Xen не сильно уступает Linux по количеству инструментов. Но существует практический фактор, по которому предпочтительнее становится Linux – мануалы и примеры использования Rumpkernel на данный момент есть для Linux (Ubuntu 18.04), а для Xen нет, поэтому Linux выбран в качестве хост-платформы для оптимизации. В дальнейшем планируется оптимизация также на Xen.

2.3.2. Выбор средства измерения времени

Так как для желаемого сравнения необходима высокая точность измерения времени, важно использовать встроенные в используемое окружение средства, предназначенные для этого. Стандартными средствами измерения времени на Linux являются утилиты time и date, но, так как необходимо измерять время начиная от старта ядра операционной системы, то пришлось бы отдельно создавать механизм, отслеживающий stdout на предмет вывода сообщения о запуске ядра. Но для таких целей идеально подходит утилита ts из пакета moreutils, так как она добавляет таймстемпы времени хост-системы перед каждой строкой stdout с точностью до наносекунд с помощью clock_gettime(CLOCK$_{M}ONOTO$

3. Реализация

3.1. Rumpkernel

В ходе изучения основ взаимодействия с операционной системой Rumpkernel были найдены и исправлены несколько ошибок в сценариях сборки, как правило связанных с тем, что ее активная поддержка завершилась в 2015 году и спецификации используемых ею компонент изменились. Кроме этого, были исправлены различные ошибки работы на Ubuntu 18.04.4 приложения rumpctrl, предназначенного для прямой работы с использованием ядра rumpkernel и предоставленными утилитами. Его я использовал для знакомства с Rumpkernel и исследования общих принципов его работы.

3.1.1. Решение проблемы запуска

Используя в качестве хост-системы Ubuntu 18.04.4, процесс сборки Rumprun Unikernel, следуя документации [16], влечет ошибку, вызванную несоотвествием спецификаций опций gcc, отвечающих за игнорирование предупреждений его современной версии. Для исправления этой ошибки были устранены некоторые причины предупреждений компилятора, и в нескольких случаях отключены соответствующие опции.

Скрипт с исправлениями можно найти в репозитории: [18]

3.1.2. Портирование приложений

Портирование POSIX-совместимых приложений в Rumprun Unikernel происходит с помощью кросс-компиляции. Используя версию компилятора gcc для NetBSD-схожих систем, формируется исполняемый файл, который затем соединяется с ядром при помощи утилиты rumprun-bake. В процессе запуска полученного образа операционной системы на эмуляторе QEMU возникли некоторые трудности с указанием корректных опций сетевого стека, решение было найдено в документации [16], пример с httpd можно увидеть в репозитории[18].

3.2. Эксперимент

Для сравнения производительности операционных систем было исследовано время запуска httpd в стандартных конфигурациях сборки. С помощью gdb было проверено, что первая строка в выводе(”Embox kernel start” для Embox и ”rump kernel bare metal bootstrap” для Rumpkernel) происходит до начала инициализации систем. Поэтому время от вывода этой строки до старта httpd считалось временем запуска системы.

3.2.1. Условия

Хост-система: Ubuntu 18.04.4
Платформа: x86-64
Конфигурация Embox: x86/qemu
Конфигурация Rumpkernel: qemu hw-generic
Процессор: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Аппаратное ускорение(KVM) выключено.

3.2.2. Артефакты

Для того, чтобы измерить время запуска ОС на QEMU, необходимо было отделить время запуска самой QEMU от запуска ОС, поэтому был использован вышеописанный метод для нахождения времени запуска.

Для применения этого метода было необходимо запустить QEMU с ОС в фоновом режиме, передав нужные опции и присоединив утилиту ts и утилиту tee для дублирования вывода в файл, после чего ”мониторить” файл с выводом на предмет строки ”httpd”. После нахождения такой строки фоновому процессу посылался SIGKILL.

На основе такого метода были созданы bash-скрипты для автоматического запуска каждой из операционных систем в бэкграунде на QEMU, завершения их сразу после запуска httpd и подсчета среднего времени запуска.

3.2.3. Результаты

Были выполнены 10 замеров для каждой ОС.

В среднем(округленно)

Embox: 347 мс

Rumpkernel: 361 мс

3.3. Анализ и оптимизация

3.3.1. Первичные выводы

Учитывая, что httpd в среднем быстрее запускается на Embox, был сделан вывод, что дальнейшее исследование запуска Rumpkernel на данный момент не принесет значимых плодов по сравнению с непосредственным исследованием запуска Embox. Поэтому было решено перейти сразу к более глубокому исследованию процесса запуска Embox и его оптимизации.

3.3.2. Распределение времени загрузки модулей

Цель этого шага - получение распределения времени загрузки модулей Embox для выявления и оптимизации наиболее ресурсозатратных из них. В качестве такого распределения было взят результат вывода сообщений о завершении инициализации модулей с соответствующими таймстемпами. В результате было получено распределение, представленное на графиках. Таким образом, видно, что в верхней части графика есть модули, явно не требующиеся для выполнения функциональности сервера. Помимо этого, следует обратить внимание, что такие модули, как rootfs, занимают слишком много времени инициализации относительно своей функции, поэтому их код стоит рассматривать в качестве объекта для оптимизации в первую очередь.

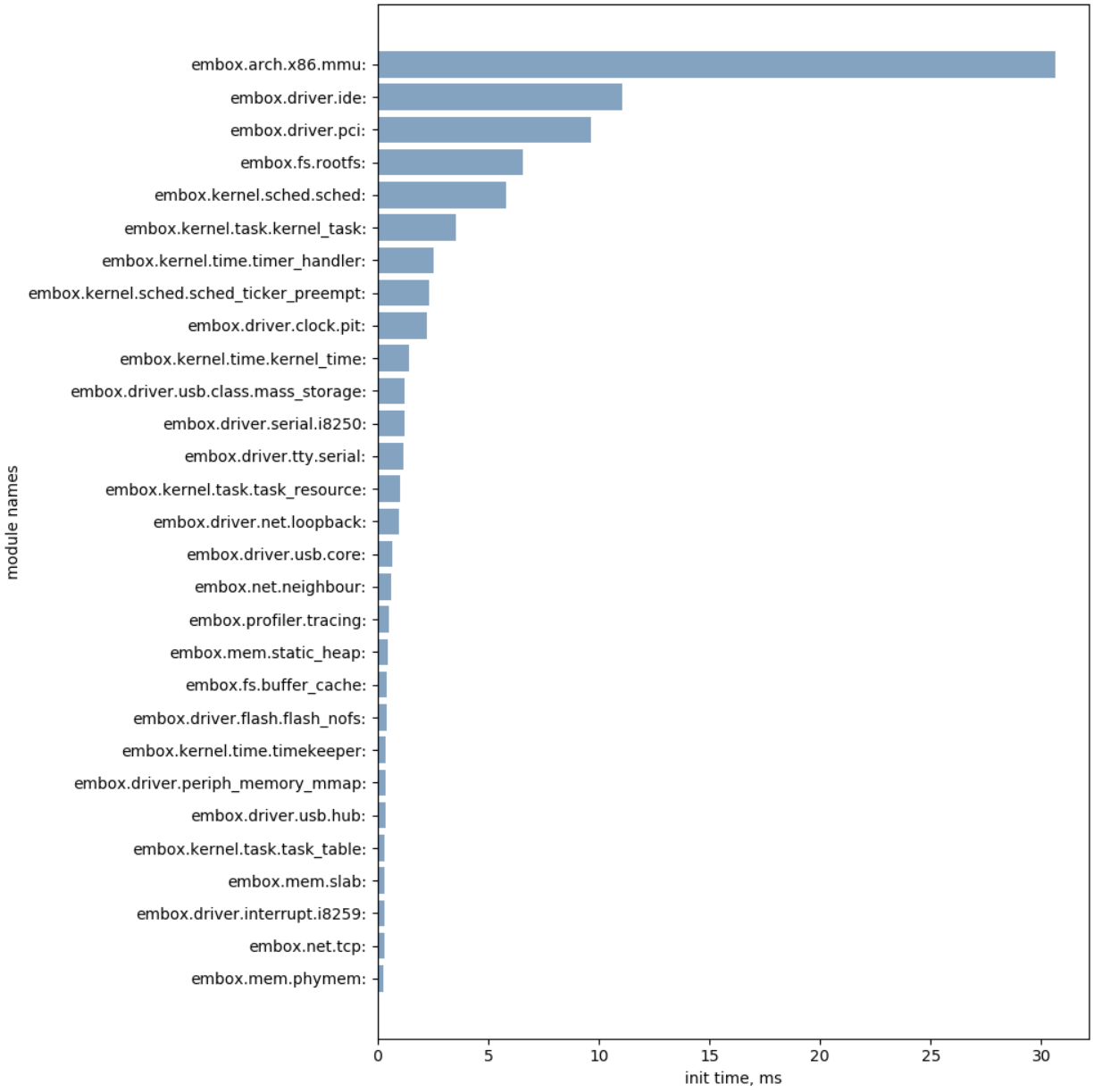
3.3.3. Изменение конфигурации Embox

Для ускорения инициализации модулей были в первую очередь отключены тесты во время инициализации, включены оптимизации во время компиляции(-O3). После изменения флагов в процессе компиляции были найдены новые несоответствия в коде(с точки зрения компилятора, логически было все верно), они были исправлены.

Далее исключены модули, не требующиеся для запуска httpd.

Для Linux к таковым относится только profiler.tracing.

Рис. 1: Распределение времени инициализации модулей Embox на QEMU

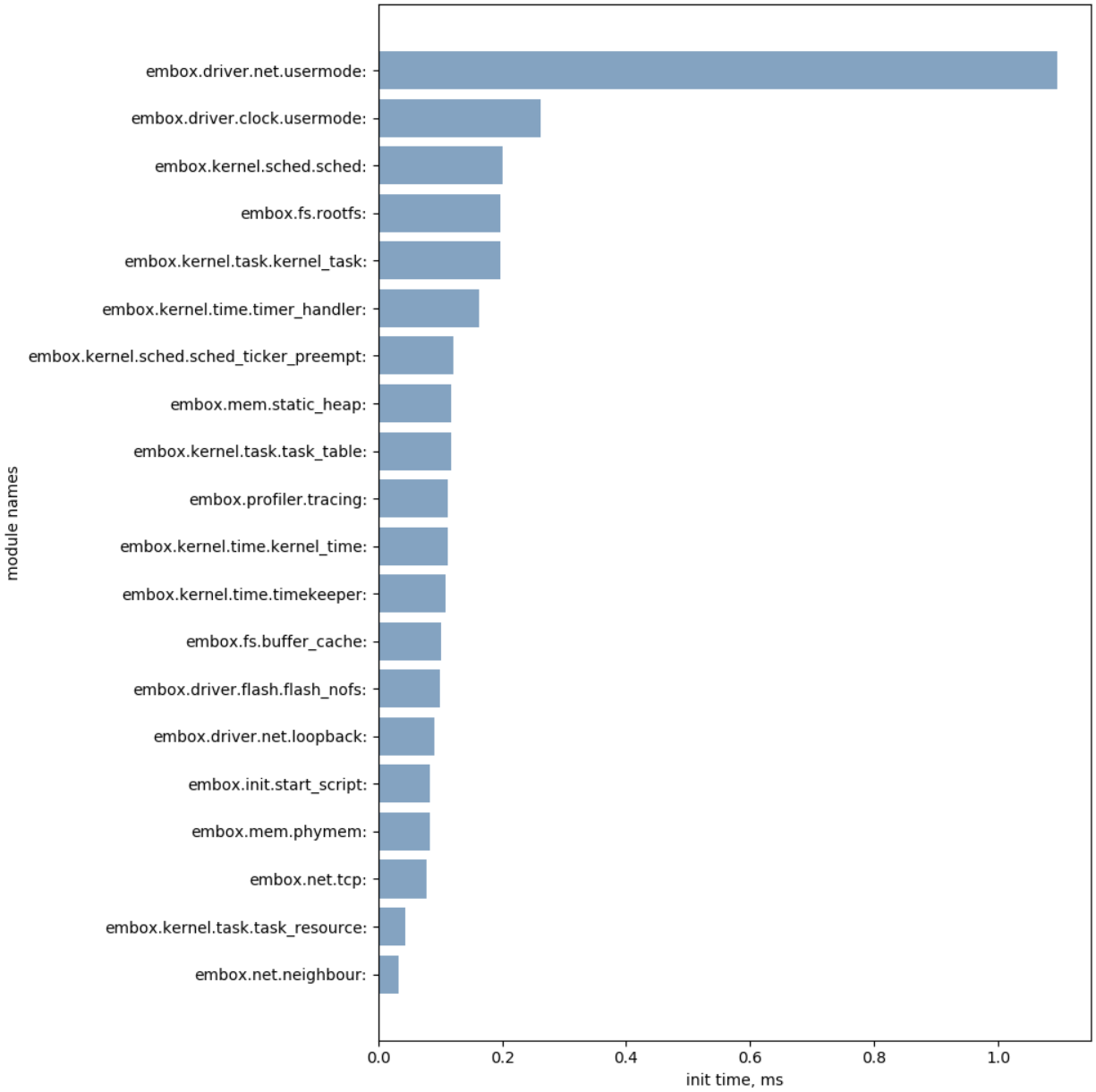


Список модулей для QEMU:

1. profiler.tracing
2. arch.x86.mmu
3. driver.usb.*

В результате, по сравнению с начальной конфигурацией среднее время запуска было уменьшено на 320 мс, сохранив функциональность.

Рис. 2: Распределение времени инициализации модулей Embox на Linux



3.3.4. Другие оптимизации

Далее было принято решение исследовать время запуска Embox на Linux с помощью утилиты perf, так как результаты оптимизаций, связанных с сокращением числа системных вызовов или более выгодным с точки зрения времени их использованием с высокой вероятностью будут переносимы на другие платформы, в том числе QEMU. Для получения профиля были использованы:

perf record -F 2000 –call-graph dwarf

perf report –call-graph

Рис. 3: Интересующая(верхняя) часть результата профиллирования Embox на Linux, полученного на старте системы

В результате исследования вывода утилиты perf было выяснено, что очень много времени уходит на системные вызовы, такие как ”сон” процесса и переключение задач, причины выяснены не были.

Заключение

В результате проделанной работы были выполнены:

1. Обзор потенциальных субъектов; в итоге кроме Embox был выбран Rumprun Unikernel

2. Анализ подходов к определению характеристики времени запуска:

   - Обоснование выбора httpd в качестве полезного приложения
   - Создание методик и инструментов для измерения и анализа времени запуска ОС
     (a) Обзор сред запуска: для сравнения была выбрана платформа QEMU, для оптимизации Linux и QEMU
     (b) Обзор средств измерения времени: для измерения времени была выбрана утилита ts(clock_gettime)

3. Сравнение времени запуска Embox и Rumprun Unikernel: было выяснено, что в начальных конфигурациях httpd в среднем запускается быстрее на Embox, чем используя Rumprun Unikernel

4. Анализ результатов: получено распределение времени загрузки модулей Embox и выделены модули, не влияющие на работу httpd

5. Оптимизация конфигураций и компонент Embox: в результате оптимизации среднее время запуска httpd на Embox было уменьшено с 383 мс в начальной конфигурации до 62.6 мс.

3.4. Перспективы

В дальнейшем необходимо произвести сравнение тех же субъектов в других условиях: выбрать другое приложение, другие конфигурации, выполнить сравнение на других машинах, платформах. Затем, можно провести подобное сравнение Embox с другими Unikernel операционными системами.

Список литературы

[1] Amazon EC2. — https://aws.amazon.com/ru/ec2/. — 2019.

[2] ClickOS. — http://sysml.neclab.eu/projects/clickos/. — 2019.

[3] CliveOS. — http://lsub.org/ls/clive.html. — 2019.

[4] DrawbridgeOS. — http://research.microsoft.com/en-us/projects/drawbridge/. — 2019.

[5] Embox. — https://github.com/embox/embox. — 2019.

[6] HaLVMOS. — http://galois.com/project/halvm/. — 2019.

[7] HermitCoreOS. — http://sysml.neclab.eu/projects/clickos/. — 2019.

[8] IncludeOS. — https://www.includeos.org/. — 2019.

[9] Linux. — https://www.linux.org/. — 2019.

[10] MirageOS. — https://mirage.io/. — 2019.

[11] OSv. — http://osv.io/. — 2019.

[12] QEMU. — https://www.qemu.org/. — 2019.

[13] Rumpkernel. — https://github.com/rumpkernel. — 2019.

[14] Unikernel. — http://unikernel.org/blog/. — 2019.

[15] Xen. — https://xenproject.org/. — 2019.

[16] rumpkernel-tutorials. — https://github.com/rumpkernel/wiki/wiki/Tutorial. — 2015.

[17] runtime.jsOS. — http://runtimejs.org/. — 2019.

[18] source-repo. — https://github.com/nikitavlaev/embox-benchmark. — 2019.