Анализ производительности сетевой подсистемы микроядерного окружения Genode

Василий Сартаков and Александр Тарасиков

ksys labs

Аннотация Микроядерные операционные системы имеют долгую историю развития, однако на сегодняшний день большинство из них остаётся экспериментальными исследовательскими проектами. В данной работе представлен опыт использования микроядра L4 Fiasco.OC и окружения Genode для построения сетевой инфраструктуры, приведен анализ ее производительности, а так же рассмотрены архитектурные особенности микроядра и программного окружения влияющие на пропускную способность сетевой подсистемы.

1 Введение

Современное сетевое оборудование обладает большим функционалом, реализация которого уже давно не возможна только на аппаратном уровне. Как следствие, такое оборудование содержит в себе специализированный процессор, на котором исполняется операционная система и прикладное программное обеспечение.

Операционная система в сетевом оборудовании несет на себе основную нагрузку по обеспечении безопасности и реализации основного функционала, как следствие, для разработки *доверенных* решений операционная система должна обладать повышенными средствами защиты от проникновения и отказов.

Микроядерные операционные системы - это такие операционные системы, в которых большая часть функциональности ядра реализована в виде отдельных сервисов. Подобные системы находятся в разработке с 1970-х годов; некоторые проекты достигли коммерческого успеха и применяются в промышленности. Однако, особенности архитектуры микроядра могут накладывать ограничения на производительность и границы применимости таких систем [5].

Основные механизмы обеспечения безопасности в микроядерной среде - выполнение всех процесов в депривелигированном (пользовательском) режиме и использование системы межпроцессного взаимодействия вместо прямого обращения к памяти другого процесса[6]. Большая часть функциональности, такой как драйвера устройств, файловые системы, управление памятью, реализуется в виде независимых процессов, выполняющихся в пространстве пользователя. Компрометация одной из подсистем ядра не дает атакующему доступ в привелигированный режим, в то время, как в получивших большее распространение монолитных ОС, где все системные компоненты выполняются в пространстве ядра (режиме супервизора), такая угроза есть.

В данной статье представлен опыт создания сетевого оборудования на основе экспериментального микроядра Fiasco.OC в окружении Genode. Используемое микроядро, с одной стороны, обладает высоким потенциалом с точки зрения безопасности, поскольку вынесение критического функционала из ядра в пространство пользователя, а так же жесткая изоляция компонентов [1] уменьшают количество векторов атак и повышает отказоустойчивость. С другой стороны, изоляция компонентов ядра приводит к интенсивному межпроцессному обмену, что может негативно влиять на производительность сетевого оборудования.

Для оценки производительности такого оборудования был разработан прототип аппаратно-программного комплекса, решающего задачу изоляции сетевый сервисов при помощи паравиртуализации. В качестве гипервизора

[1] Под изоляцией подразумевают использование раздельных адресных пространств и использование принципа наименьших привилегий в выделении ресурсов

было использовано экспериментальное микроядро Fiasco.OC, виртуальные машины реализованы на основе паравиртулизированного ядра L4Linux[4].

Результирующие тесты производительности системы показали деградацию пропускной способности сети в сравнении с системами построенными на монолитно-модульном ядре Linux. Для выявления причин деградации был разработан набор программного обеспечения для профилирования ядра и окружения. Этот набор инструментов позволил выявить компоненты программного окружения, вносящие наибольший вклад в падение производительности сетевой подсистемы, а также отличия в поведении микроядра Fiasco.OC на одно- и многопроцессорных системах.

Статья имеет следующую структуру: В первой главе представлена архитектура экспериментального аппаратно-программного комплекса, а так же результаты нагрузочного тестирования. Во второй главе описана методология профилирования и ее результаты. В третьей главе представлен анализ профилирования, а в четвертой главе приведены рассуждения о методах повышения производительности. В заключительной, пятой главе, подводятся итоги работы и формулируются планы на дальнейшую работу.

2 Аппаратано-программный комплекс «Шлюз»

2.1 Функциональное назначение

Гарантированное разделение ресурсов в вычислительных системах - одна из основных задач информационной безопасности. При этом есть необходимость как ограничить доступ к вычислительным ресурсам внутри одной машины, так и контролировать потоки данных между различными сетевыми устройствами.

При проектировании корпоративной локальной сети, необходимо изолировать безопасный сегмент сети (сетевое хранилище и автоматизированные рабочие места работников, обрабатывающих внутрикорпоративные документы) и сетевые устройства, имеющие доступ в интернет или другие публичные сети. Возникает задача контроля входящих сетевые соединения для того, чтобы блокировать трафик от недоверенных узлов. Кроме того, требуется производить мониторинг сетевого трафика для обнаружения специальным образом сформированных сетевых пакетов, направленных на повышение привелегий злоумышленника путем эксплуатации уязвимостей в программном и аппаратном обеспечении.

Зачастую требуется предоставить доступ к части внутрисетевых ресурсов. Например, внутри безопасной локальной сети пользователям могут быть доступен сервис сетевого хранения данных, реализованный в виде сетевого диска. Если определенные файлы необходимо сделать доступными из внешней сети, может использоваться веб-сервер для выдачи индивидуальных файлов. Помимо возможности проведения дополнительной авторизации пользователей средствами веб-интерфейса, такое решение скрывает от потенциальных злоумышленников физическую топологию внутрикорпоративной сети.

Ряд приложений требует передачи конфиденциальных данных по незащищенному (публичному) каналу. Для организации безопасного обмена данными используются туннели с шифрованием трафика. При этом требуется гарантировать, что ключи, использующиеся для шифрования, не будут скомпрометированы. Поэтому необходим механизм разделения вычислительных процессов внутри узла сети, реализующего туннель.

Рассмотренные выше задачи решаются использованием сетевого шлюза, то есть, сетевого устройства, имеющего связь с безопасным и небезопасным сегментами сети, и ограничивающего обмен данными между ними. Схема топологии сети с применением сетевого шлюза приведена на рисунке 1. Далее представлена реализация сетевого шлюза с использованием микроядера Fiasco.OC.

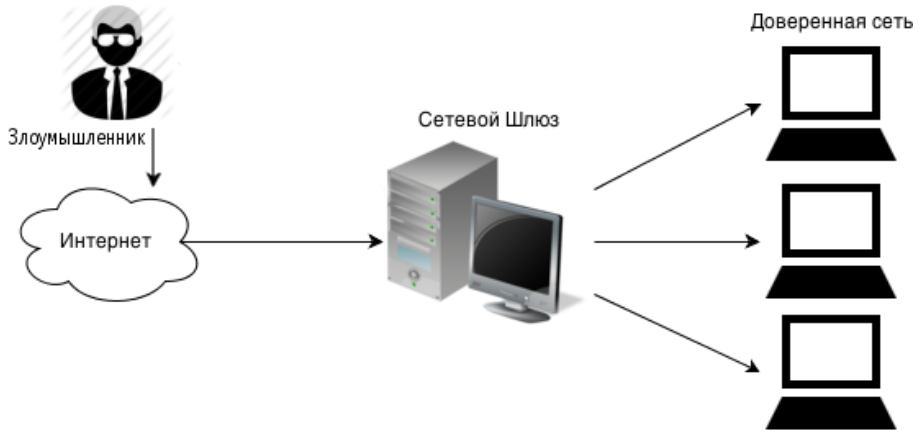

Рис. 1. Схема сети с использованием шлюза

2.2 Реализация

Рассмотрим одну из возможных реализаций описанного выше сетевого шлюза. В качестве аппаратной части комплекса используется сервер с двумя сетевыми интерфейсами, один из которых обеспечивает связь с безопасными, а другой - с небезопасными сегментами сети. Задачи фильтрации и маршрутизации трафика решаются при помощи виртуальных машин. Для обмена данными между виртуальными машинами может использоваться как виртуальный сетевой интерфейс, так и криптографический сервис, обеспечивающий конфиденциальность передаваемых по незащищенному сегменту сети данных. Программная часть реализована при помощи микроядра Fiasco.OC, программной среды Genode и паравиртуализованного ядра L4Linux.

Поскольку стояла задача оценить производительность полностью микроядерного окружения, в котором драйвера сетевой подсистемы реализованы в пространстве пользователя, были исключены решения на базе гибридных ядер ОС. Кроме того, к используемой комбинации ядра ОС и программного окружения предъявлялись следующие требования:

– Открытый исходный код, так как предполагалась доработка системы и последующее распространение изменений.
– Поддержка микропроцессорной архитектуры X86 и многопроцессорных (SMP) систем.
– Поддержка современного оборудования, в частности, наиболее распространённых моделей сетевых карт.

Этим требованиям удовлетворяют микроядра семейства L4, такие, как OKL4, Fiasco, Fiasco.OC. Было использовано ядро Fiasco.OC, как наиболее активно развиваемое (новые версии выходят раз в несколько месяцев, в то время как последний публичный релиз OKL4 был несколько лет назад) и официально поддерживаемое окружением Genode.

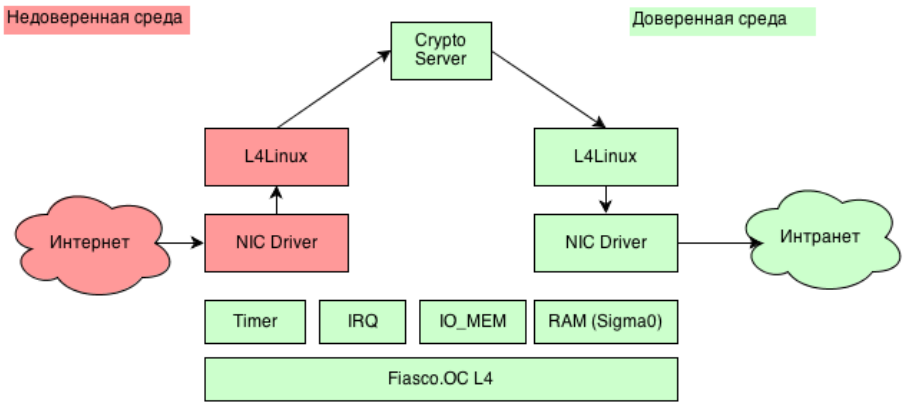

Рис. 2. Архитектура сетевого шлюза с использованием микроядра

Программная среда Genode - это набор системных сервисов (драйверов устройств и файловых систем) и библиотек, предназначенных для разработки программного обеспечения для микроядерной ОС. Ряд библиотек реализуют программный интерфейс POSIX, позволяя переносить некоторые приложения из ОС Linux посредством перекомпиляции. Кроме того, Genode содержит библиотеку DDEKit[7], позволяющую использовать ряд драйверов устройств из других ОС (таких, как Linux и iPXE). Одним из сервисов, использующих библиотеку DDEkit, является DDE_ipxe, реализующий функциональность драйвера сетевого интерфейса. Таким образом, драйвер

сетевого интерфейса представляет собой программу, выполняющуюся в пространстве пользователя, тем самым изолированную от других сервисов. В случае, если будет обнаружена уязвимость в этом драйвере, позволяющая злоумышленнику внедрить свой код в сервис, не произойдет компрометации других программ, в частности, приложений, выполняющихся в виртуальных машинах.

Виртуальные машины реализованы при помощи L4Linux. L4Linux - это адаптированное ядро Linux для работы в депривилегированном режиме (пространстве пользователя) в качестве сервиса среды Genode. L4Linux позволяет использовать программы для ОС Linux без модификации за счет двоичной совместимости. Помимо сокращения затрат времени на адаптацию программного обеспечения для работы в микроядерной среде, это позволяет использовать программы, которые невозможно адаптировать ввиду отсутствия исходных кодов, например, значительная часть коммерческого программного обеспечения.

2.3 Модель угроз

Основной вектор атаки на сетевые устройства под управлением монолитных ОС типа Linux и FreeBSD - это эксплуатация уязвимостей в одном из сервисов, доступных извне (например, HTTP сервер, используемый для конфигурации или вывода информации) для повышения привилегий. Ряд атак позволяет злоумышленнику повысить уровень доступа в пределах одного сервиса, другие направлены на получение полного доступа к ядру ОС (root expoits). Большое количество атак для повышения привилегий использует ошибки в ПО, связанные с переполнением буфера [3]. Атака с переполнением буфера заключается в возможности перезаписи переменных в программе из-за ошибок обработки пользовательского ввода. Данный класс уязвимостей возможен в связи с тем, что для хранения кода программ и пользовательских данных используется одна и та же область памяти. Кроме того, для передачи параметров в системные вызовы зачастую используется разделяемый между пространством ядра и пространством пользователя регион памяти, что позволяет получить доступ к структурам ядра, если в коде обработки системного вызова отсутствуют или некорректно реализованы проверки размера передаваемых данных.

В микроядре Fiasco.OC взаимодействие между процессами реализовано через систему обмена сообщениями. Сообщения передаются через специальный регион памяти фиксированного размера (UTCB, Userspace Thread Control Block). При передаче данных от процесса к процессу данные копируются из UTCB первого процесса в UTCB второго, что, с одной стороны, защищает от утечек данных через указатели на структуры ядра, а с другой, может увеличить время затрачиваемое на системные вызовы. Важно заметить, что эта мера поддерживающая целостность данных при осуществлении межпроцессных вызовов, тем самым не позволяя злоумышленнику, получившему контроль над каким-либо сервисом, атаковать сервисы,

связанные с ним, но не гарантируют невозможность проведения атаки на отдельный сервис.

Одной из угроз вляется атака через устройства, поддерживающих DMA. Технология DMA (Direct Memory Access) позволяет устройству обращаться к произвольной области физической памяти. Защища от таких атак не зависит от типа ядра и реализуется при помощи IOMMU (Input/Output Memory Management Unit) - аппаратного блока виртулизирующего физическую память с которой работает DMA устройство.

Другим типом угроз является атака на драйвер устройства. Выявление и использование уязвимости в сетевой подсистеме не дает доступа злоумышленнику к структурам ядра поскольку сетевая подсистема исполняется в режиме пользователя и изолированная от других компонентов.

2.4 Анализ производительности

Ключевой характеристикой аппратно-программного комплекса являлась производительность. Как упоминалось ранее, микроядерная архитектура способствует повышению защищенности системы, но, в то же время, может способствовать деградации производительности. При анализе производительности мы использовали ОС Linux в качестве эталонной, исполняемой на таком же оборудовании без использования гипервизора.

Для измерения производительности сети были использованы два компьютера - рабочая станция разработчика (далее - PC Host) и сервер, представляющий собой аппаратную часть разрабатываемого комплекса (далее SRV). Оба компьютера оснащены сетевым контроллером Intel 82579LM с пропускной способностью 1Гб/с. Для проведения тестов использовалась программа *netperf* [1].

Были проведены тесты приема и передачи данных. В первом случае один экземпляр программы netperf запускался в режиме сервера внутри среды L4Linux, а другой - в режиме клиента на рабочей станции под управлением Linux (на графике данная конфигурация обозначена как «SRV Host - PC Client»). Во втором случае («PC Host - SRV Client») в L4Linux netperf запускался в режиме клиента. Для рабочей станции использовался дистрибутив Fedora Linux 18 с версией ядра Linux 3.8; при запуске на сервере в среде Genode использовалось ядро L4Linux версии 3.5, в качестве корневой файловой системы также использовался образ дистрибутива Fedora.

Результаты тестирования приведены на графике ниже (Рис. 3). Можно выделить следующие особенности:

- Максимальная пропускная способность сети с использованием системы L4Linux в среде Genode в 1.75 раз ниже, чем без использования паравиртуализации
- Показатели скорости передачи данных отличаются для входящего и исходящего трафика. Скорость приёма данных почти в 10 раз ниже, чем без использования паравиртуализации.

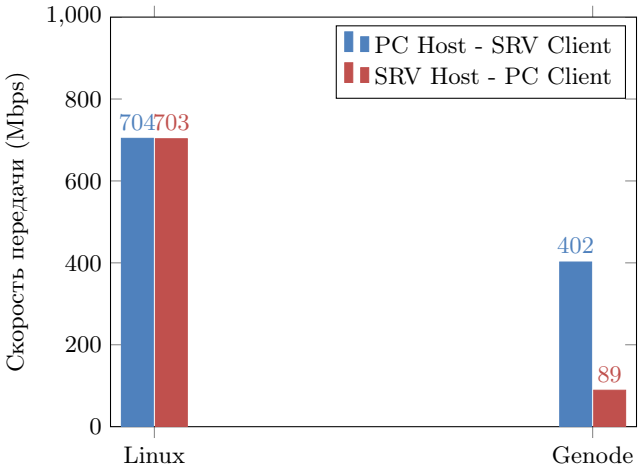

Рис. 3. Сравнение производительности сетевой подсистемы

3 Профилирование сетевого стека

Высокая деградация производительности полученной системы говорит о недостатках архитектуры. Дальнейшие эксперименты по анализу производительности системы осуществлялись на одном экземпляре L4Linux который непосредственно обращался к драйверу сетевого адаптера *DDE_iPXE*.

3.1 Инструментарий профилирования

Программная среда Genode не предоставляет средств для профилирования кода. Была рассмотрена возможность портирования инструментов, используемых в других ОС, таких, как *gprof* или *DTrace*. Одним из препятствий к портированию данных утилит стало отсутствие в библиотеках, реализующих уровень совместимости со стандартом POSIX, некоторых системных вызовов, например, *profil* и *mcount*, необходимых для реализации инструментирующего профилировщика. Кроме того, сложность представляет организация доступа из профилировщика к объектам ядра ОС и аппаратным счётчикам производительности без внесения изменений в конфигурацию сервисов и нарушения принципа изоляции процессов в микроядерном окружении.

Ввиду отсутствия средств измерения производительности, в участки кода драйвера сетевого интерфейса были вручную расставлены счетчики тиков системного таймера (получаемых инструкцией rdtsc), затрачиваемых на выполнение интересующей нас процедуры. Сообщения выводились в журнал, который передавался на компьютер разработчика через последовательный порт UART.

При использовании пакетов большего размера было обнаружено, что из-за низкой скорости вывода данных через UART, добавление отладочных

сообщений замедляет работу исследуемой программы. Для решения этой задачи был реализован сервис, сохраняющий отладочный журнал в оперативную память. Был использован существующий драйвер *ram_fs* в среде Genode, который реализует файловую систему в оперативной памяти. По окончании тестирования журнал копировался с тестового компьютера через HTTP интерфейс, предоставляемый сервером *lighttpd*, запущенным в среде Genode.

В приведенной ниже таблице 1 представлено среднее время выполнения функций сетевого стека (в тиках процессора).

Процедура	Количество тиков	Файл
dde_kit_sem_up	13	lib/nic.c [dde_ipxe]
dde_kit_sem_down	59	lib/nic.c [dde_ipxe]
memcpy	23	lib/nic.c [dde_ipxe]
get_acked	46	nic/component.h [Nic]
submit_packet	190-160K	nic/component.h [Nic]
packed_descriptor	10	nic/component.h [Nic]
get_packet	8	lib/l4lx/genode_net.cc [L4Linux]
acknowledge_packet	254	lib/l4lx/genode_net.cc [L4Linux]
submit_packet	65	lib/l4lx/genode_net.cc [L4Linux]
memcpy	25	lib/l4lx/genode_net.cc [L4Linux]

Таблица 1. Время выполнения процедур сетевого стека

3.2 Анализ результатов профилирования

Как следует из таблицы 1, наиболее затратная операция - *submit_packet*, которая помещает пакет в циклический буфер и делает IPC запрос к другому процессу. При тестировании программой *netperf* было обнаружено, что при длительной передаче данных время выполнения операции *submit_packet* спорадически увеличивается на несколько порядков. Было сделано предположение, что причинами такого поведения могут быть как особенности реализации драйверов устройств, так и алгоритмов управления процессами и памятью в ядре ОС.

Драйвера устройств

Одной из причин несимеетричной производительности сетевой подсистемы при приеме и передаче данных может являться реализация обработки аппаратных прерываний в микроядре Fiasco.OC. В Genode не реализована поддержка прерываний, инициируемых сообщениями (MSI-X) для шины PCI. Это не позволяет назначить обработчик для прерывания, сигнализирующего о наличии входных данных на сетевом интерфейсе, и приходится

использовать прерывание от шины PCI, которое может быть разделено с другими устройствами, что может приводить к увеличению времени обработки входящего трафика. Кроме того, драйвер контроллера прерываний IO APIC на некоторых системах, оставляет контроллер в состоянии, в котором тот остался на момент загрузки, не реализуя включение/выключение источника прерываний, что осложняет отладку программного комплекса.

Управление памятью

В процессе тестирования программа *netperf* последовательно увеличивает размер сетевых пакетов. При длительном тестировании наблюдается большой разброс времени выполнения операции *submit_packet*, в связи с чем было сделано предположение, что драйвер сетевого интерфейса тратит много времени в ожидании выделения памяти для очередного пакета. С целью проверки этого предположения были увеличены в 8 раз размеры буферов для входящих и исходящих данных. Поскольку увеличение размеров буферов не внесло изменений в поведение системы, необходимо было провести исследование поведения алгоритма выделения памяти. Для управления памятью в сетевой подсистеме Genode используется алгоритм SLAB[2], который содержит несколько списков блоков памяти фиксированного размера (1Кб, 2Кб, 16Кб и т.д.). Память для пакетов с данными организована в виде кольцевого буфера, где пакет, освобожденный одним из сервисов (например, драйвером сетевого адаптера DDE_ipxe), попадает в очередь свободных пакетов другого сервиса (драйвер виртуального сетевого интерфейса L4Linux). При передаче региона памяти в процесс другого сервиса выполняется операция освобождения региона (unmap), и операция выделения памяти в новом процессе. При конфигурации микроядра Fiasco.OC для поддержки многопроцессорного режима (SMP), реализация многих алгоритмов и структур данных, в том числе примитивов синхронизации (мьютексов), отличается от однопроцессорной версии. Кроме того, при использовании нескольких процессоров потоки, выполняющие системные функции (обработку прерываний, ввод-вывод) и пользовательские задачи выполняются одновременно. Для определения влияния данных различий на производительность системы, было проведено тестирование в двух режимах: при включении всех ядер процессора поддержки SMP в Fiasco.OC (время выполнения теста составило 24929000 мкс), и при использовании одного ядра и отключения SMP (146000 мкс). При использовании одного ядра процессора, разница между скоростью приема и передачи данных уменьшается. В многопроцессорном режиме планировщик проводит значительную часть времени (до 30%) в режиме простоя (idle thread) вместо того, чтобы передавать управление потокам приложений. Изучение причин такого поведения планировщика и сравнение с другими микроядрами, поддерживаемыми средой Genode является задачей для дальнейших исследований.

4 Предлагаемые решения

Для минимизации задержек при обработке сетевых данных были предложены несколько способов доработки сетевой подсистемы:

- Использовать регион разделяемой памяти (Dataspace в терминологии L4/Genode) вместо буферизации в каждом процессе. Но такое решение противоречит принципам безопасности микроядерных систем, так как в случае компрометаци драйвера сетевого интерфейса злоумышленник потенциально получает доступ к памяти всех процессов, использующих сеть.
- Предоставить паравиртуализированным экземплярам L4Linux доступ к физическому сетевому адаптеру. Для обеспечения доступа к памяти и регистрам PCI устройств был реализован драйвер виртуальной PCI шины для L4Linux, однако проблема обработки прерываний в ACPI системах не позволяет провести полноценное тестирование решения. Кроме того, очевидные минусы - невозможность использования одного сетевого интерфейса различными сервисами.
- Минимизировать количество потоков выполнения (threads), занимающихся обработкой данных. Синхронизация доступа к критическим секциям может занимать значительную часть времени обработки сетевого пакета, как можно видеть в таблице 1. Некоторые подсистемы среды Genode (в частности, DDE_kit) переделаны для выполнения в однопоточным режиме. Для задач, производительность которых ограничена операциями ввода-вывода, использование многопоточной архитектуры зачастую увеличивает латентность.
- Реализовать поддержку прерываний MSI-X и контроллера IO APIC и провести анализ влияния используемого источника прерываний (от устройства посредством MSI-X и через шину PCI) на время обработки входящих данных.

5 Заключение

Был разработан рабочий прототип доверенного сервера, обеспечивающего изоляцию приложений при помощи паравиртуализации в среде Genode/L4Linux. В ходе тестирования было обнаружено падение производительности сетевой подсистемы, не позволяющее использовать полученное программное решение в качестве непосредственной замены ОС Linux. Детальный анализ прохождения нагрузочных тестов выявил ряд недостатков в дизайне системы, связанных с драйверами устройств и механизмом выделения памяти в многопроцессорной системе. Дальнейшая работа будет посвящена оптимизации драйверов и реорганизации многозадачности.

Список литературы

1. Netperf homepage. `http://netperf.org`.

2. Jeff Bonwick et al. The slab allocator: An object-caching kernel memory allocator. In *USENIX summer*, volume 16. Boston, MA, USA, 1994.
3. Crispin COwan. Buffer overflows: Attacks and defenses for the vulnerability of the decade. 2000.
4. TU Dresden. L4linux-running linux on top of l4.
5. Hermann Haertig et al. The performance of u-kernel-based systems. In *ACM Simposum on Operating Systems Principles*, volume 16. Saint-Malo, France, 1997.
6. Jochen Liedtke. Toward real microkernels. *Communications of the ACM*, 39(9):70–77, 1996.
7. Hannes Weisbach. Ddekit approach for linux user space drivers. 2011.