Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Московский физико-технический институт
(государственный университет)»

Факультет радиотехники и кибернетики

Кафедра теоретической и практической информатики

Исследование производительности паравиртуализации устройств на примере камеры

Выпускная квалификационная работа
(бакалаврская работа)

Направление подготовки: 03.03.01 Прикладные математика и физика

Выполнил:
студент 211 группы _____ Кайда Александр Васильевич

Научный руководитель:
_____ Симановский Андрей Юрьевич

Москва 2016

Оглавление

Аннотация

В данной работе исследуется производительность паравиртуализованных устройств на гипервизорном виртуализационном решении. Основной целью является проверка предположения о том, что паравиртуализация может быть эффективной техникой виртуализации для устройств с большим объёмом передаваемых данных. В качестве примера такого устройства используется видеокамера.

1. Введение

1.1. Обзор предметной области

Виртуализация компьютерных систем является важной и востребованной областью информационных технологий. В наше время активно развивается и используется множество виртуализационных решений. Использование таких решений позволяет разделить компьютерные ресурсы между несколькими изолированными экземплярами операционных систем. Такими ресурсами могут выступать процессорное время, память, диск или другие устройства.

Следует выделить два основных типа виртуализации компьютера: уровня платформы и уровня операционной системы (контейнерная виртуализации). Рассмотрим виртуализацию уровня платформы, основанную на гипервизоре.

В дальнейшем будет использоваться такое понятие как виртуальная машина. Это такое окружение, которое представляется гостевой ОС реальной аппаратной платформой. Но на самом деле этот эффект достигается эмуляцией оборудования, выполняемой гипервизором. Эмуляция должна достаточно точно имитировать поведение оборудования для нормального функционирования гостевой ОС.

Гипервизор — это программа, позволяющая на одном хост-компьютере запускать несколько операционных систем, называемых гостевыми. Название это возникло из того, что по уровню привилегий гипервизор находится на уровень выше супервизора (системы, управляющей запуском обычных пользовательских приложений). Гипервизор предоставляет гостевым операционным системам виртуальную аппаратную платформу, а так же обеспечивает изоляцию операционных систем друг от друга, защиту и безопасность, разделение и управление ресурсами компьютера. Гипервизор также может предоставлять средства связи между гостевыми операционными системами, например внутреннюю компьютерную сеть.

Гипервизоры разделяют на два типа: запускаемые на аппаратуре как операционные системы (“bare metal”) и запускаемые под обычной операционной системой. Примером гипервизора первого типа является VMware ESX Server, второго — Parallels Desktop. При использовании гипервизора второго типа низкоуровневые операции с устройствами выполняются основной операционной системой.

Существуют различные методы виртуализации процессора и памяти компьютера.

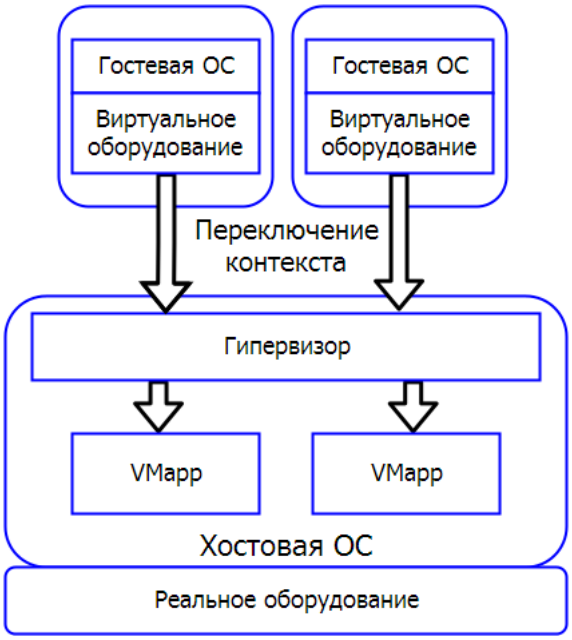

Рис. 1. Гипервизор второго типа

Наиболее эффективной на данный момент является аппаратная виртуализация. В качестве примера приведём технологию Intel VT-x и её расширение EPT, которые и будут использованы в данной работе.

Технология аппаратной виртуализации процессора заключается в наличии нового режима исполнения процессора, называемом VMX (virtual machine extension). Гипервизор исполняется в VMX root mode и обладает полными привилегиями, а гостевая ОС исполняется в VMX non-root mode. В этом режиме некоторые привилегированные команды не исполняются, вместо этого происходит выход в гипервизор и выполняется обработчик, который эмулирует проблемную инструкцию. Таким образом, гость не имеет доступа к реальному оборудованию и всей физической памяти. Так же происходит виртуализация оперативной памяти, данная технология носит называние Nested Paging.

Благодаря аппаратной виртуализации скорость исполнения гостевой ОС очень близка к скорости хоста. Однако, переключение контекста VMX очень долгая операция, и её следует по возможности избегать.

1.2. Виртуализация устройств

Весьма значимой является задача виртуализации различных устройств компьютера. Для пользователя весьма важно, что бы гостевая ОС имела доступ ко всем устройствам хоста и поддерживала аналогичный функционал.

Устройства можно разделить на несколько классов: системные и периферию. К системным устройствам, без которых невозможно функционирование виртуальных машин с современными ОС, можно отнести жёсткий диск, ACPI, таймера, контроллеры прерываний и другие. К периферийным устройствам следует отнести мышь, клавиатуру, сетевую и звуковую карту, графический ускоритель и т.д.

Очевидно, что различные устройства имеют различные требования к способу виртуализации. Различается их производительность, количество передаваемых данных, временные показатели.

Следует выделить такие цели по виртуализации устройств:

- Доступ - обеспечение доступа к функции устройства является основной целью;
- Изоляция - работа разных гостей и хоста с устройством не должны влиять друг на друга;
- Скорость - некоторые устройства имеют чёткие требования к производительности и временным задержкам, которые необходимо соблюдать для правильной работы;
- Мультиплексирование - возможность использования одного устройства несколькими виртуальными машинами и хостом;
- Мобильность - независимость виртуальной машины от аппаратного обеспечения хоста.

1.2.1. Эмуляция

Эмуляция является базовой техникой виртуализации устройств компьютера. Заключается в программном управлении памятью, портами и прерываниями виртуальной машины с целью максимально точно имитировать поведение реального оборудования. Таким образом гостевая ОС выполняется в виртуальной машине как на настоящем оборудовании.

Для эмуляции может использоваться реальное устройство, тогда стоит задача разделения ресурсов этого устройства между несколькими виртуальными машинами. В качестве примера можно привести жёсткий диск, который является SATA устройством для гостевой ОС, но при этом упакован в файл виртуального жёсткого диска на файловой системе хоста.

Основным достоинством данного подхода является его универсальность. Большинство современных ОС поддерживают стандартное оборудование и, соответственно, могут работать с эмулированными устройствами. Чаще всего именно так реализуют важные системные и пользовательские устройства: жёсткий диск, сетевую, звуковую и графические карты.

1.2.2. Паравиртуализация

Основная идея паравиртуализации заключается в изменении гостевой ОС таким образом, что бы коммуникации с гипервизором происходили посредством специальных механизмов и API. Это позволяет упростить интерфейс взаимодействия и отказаться от неэффективных операций по эмуляции реального оборудования. В некоторых случаях это позволяет расширить функционал гостевой ОС за счет дополнительных функций взаимодействия с пользователем. Например: общий с хостом буфер обмена, Drag'n'Drop файлов и т.д.

Изменения гостевой ОС могут быть как новыми драйверами, так и приложениями уровня пользователя, в зависимости от требований.

Например, в сравнении с эмуляцией USB устройства это позволяет избежать программной упаковки данных в отдельные пакеты, а так же жёстких временных ограничений протокола. Следует отметить, что в отдельных случаях это позволяет повысить пропускную способность обмена данными с устройством.

Таким образом, мы можем выделить характерные отличия паравиртуализации от эмуляции: повышенная производительность и необходимость в специальных драйверах в гостевой ОС.

1.2.3. Прокидывание

Данная техника позволяет подключить устройство напрямую к виртуальной машине программными средствами. С помощью специального драйвера на хосте запрещается доступ к устройству, и все коммуникации осуществляются непосредственно с гостевой ОС. Чаще всего необходимо дополнительное копирование передаваемых между хостом и гостем данных. Примером использования такой техники могут быть USB устройства, при

коммуникации с которыми все пакеты данных получаются драйвером на хосте и отправляются в гостевую систему.

Производительность является сильной стороной этой техники. Однако невозможность разделения используемого устройства и сложность реализации значительно ограничивают применимость прокидывания устройств.

1.2.4. Аппаратная виртуализация устройств

Технология аппаратной виртуализации устройств Intel VT-d позволяет использовать устройства на шине PCI (и подобных ей) в гостевой ОС напрямую. При этом гость может работать с устройством с помощью стандартных драйверов. Это реализовано с помощью специального устройства управления памятью ввода-вывода (IOMMU), которое осуществляет трансляцию адресов DMA. Таким образом, драйвер гостевой ОС использует виртуальные адреса DMA, которые отображаются на актуальную физическую память, используемую устройством.

Данная технология является аппаратной реализацией техники прокидывания устройства, что приводит к схожим проблемам. Из положительных сторон этого решения следует отметить максимальную производительность передачи данных. Однако недостатки весьма существенны: требуется аппаратная поддержка этой технологии на процессоре, материнской плате и на самом устройстве. Кроме того, устройство может использоваться только в одной ОС.

1.3. Постановка задачи

Некоторые устройства могут иметь уникальные аппаратные реализации, но при этом предоставлять одну и ту же функцию. Такие устройства невозможно присвоить напрямую гостевой ОС, однако их можно виртуализовать с помощью эмуляции или паравиртуализации. Примером такого устройства является видеокамера (веб-камера). На персональных компьютерах она реализована чаще всего как USB устройство, однако не всегда поддерживается стандартными драйверами. На современных планшетах с платформой x86 камера может быть реализована как сложное составное PCI устройство с закрытыми интерфейсами.

Необходимо отметить, что для нормального функционирования камеры требуется

высокая скорость передачи данных. Дополнительным важным условием в данной работе является единая гостевая ОС.

Таким образом, при выборе техники виртуализации было решено использовать паравиртуализацию. Основным вызовом для успешной работы камеры стала пропускная способность интерфейса передачи данных от хоста к гостю, которая и стала основной частью исследования.

Целью данной работы стало измерение производительности паравиртуализации устройств и исследование способов её повышения.

2. Решение

2.1. Паравиртуализация камеры

Программный комплекс, реализующий паравиртуализацию камеры можно разбить на три основные части:

- **Гостевая библиотека** - ответственна за подключение камеры к гостевой ОС, обмен данными через тулгейт. Может быть выполнена в виде драйвера;
- **Тулгейт** (toolgate)- универсальный компонент, осуществляющий передачу данных и управляющих команд из гостевой ОС в гипервизор и наоборот;
- **Работа с устройством на хосте** - включает в себя подключение к реальной камере посредством стандартного API и управление ею. Реализована в виде отдельной библиотеки.

В то время как тулгейт является вспомогательным компонентом, осуществляющим транспортную функцию, гостевая и хостовая библиотеки тесно взаимодействуют и сильно связаны между собой функционально.

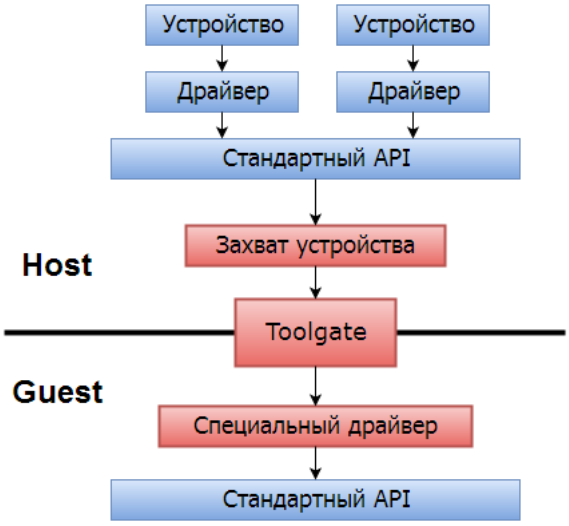

Рис. 2. Общая схема паравиртуализации устройства

Заметим, что в качестве хостовой ОС используется OS X 10.11, гостевая ОС - Android 5.1, а платформой виртуализации выступает Parallels Desktop.

2.1.1. Гостевая библиотека

Использование Android в качестве гостя значительно влияет на гостевую часть механизма. Стек камеры в данной ОС сильно отличается от стека камеры, используемого в Windows, и даже в Linux.

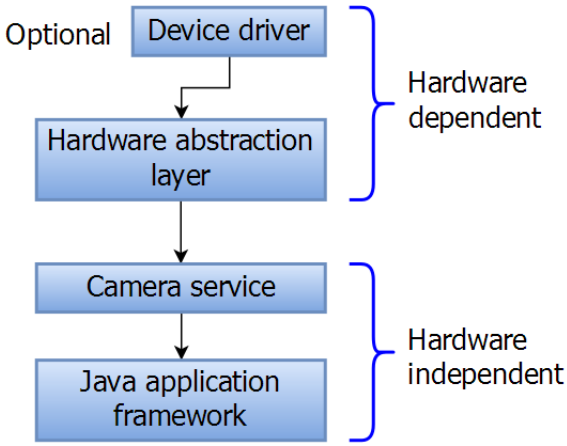

Рис. 3. Стек камеры в Android

Важным является тот факт, что помимо драйвера уровня ядра производитель камеры должен поставлять библиотеку пользовательского уровня, обеспечивающую уровень аппаратной абстракции (Hardware Abstraction Level). Таким образом, нет никаких жёстких требований к архитектуре и API используемого драйвера. Более того, в случае паравиртуализованной камеры даже удобнее использовать библиотеку пользовательского уровня для подключения к стеку камеры, что и было сделано.

Использование библиотеки пользовательского уровня вместо драйвера позволяет повысить надёжность решения и избежать критических ошибок, чреватых падением системы. Более того, это открывает доступ к стандартным библиотекам и упрощает встраивание в систему.

Опишем основные компоненты библиотеки HAL с использованием ООП:

CameraFactory - в библиотеке имеется глобальный объект этого класса. Отвечает за контроль над всеми камерами, и открытие устройств камеры.

CameraHardware - класс конкретного устройства, реализует требуемый интерфейс слоя абстракции, выполняет основные действия по управлению камерой.

PVCamera - дополнительный класс, отвечающий за обработку низкоуровневых запросов к камере и передачу данных через тулгейт.

Классы CameraFactory и CameraHardware предоставляют вышестоящему сервису набор обратных вызовов в глобальных структурах, по которым происходят все запросы к слою аппаратной абстракции камеры.

Опустив излишние детали, основной цикл работы с камерой на данном уровне выглядит следующим образом:

1. Опрос объекта класса CameraFactory на предмет подключённых камер.
2. Открытие выбранной камеры сервисом, создание и подготовка объекта класса CameraHardware. В данный момент происходит подключение к камере на хосте, и опрос её настроек.
3. Получение допустимых настроек камеры и выбор требуемых.
4. Старт потокового захвата кадров. При этом в объекте камеры создаётся поток, который через равные промежутки времени опрашивает хостовый интерфейс и получает готовые буфера с изображением.
5. Остановка захвата кадров.
6. Закрытие объекта камеры и освобождение ресурсов.

Отдельно остановимся на обработке полученных кадров объектом камеры. Необходимым действием является заполнение экранного буфера изображением требуемого формата, для чего требуется перекодирование. Благодаря этому происходит отображение видеопотока на экране. Дополнительным опциональным действием является отправка кадра сервису в определённом формате для записи. Таким образом, может потребоваться до двух перекодирований одного кадра.

В число доступных настроек камеры входит разрешение изображения, частота кадров и формат упаковки цветовых данных кадров.

Гостевая библиотека была реализована в рамках данной работы, с использованием документации и исходного кода Android.

2.1.2. Тулгейт

Тулгейт (toolgate) - программный компонент, обеспечивающий прямую передачу данных и команд между гостевой ОС и гипервизором. В виртуализационном решении VirtualBox данный компонент называется Host-Guest Communication Manager. Тулгейт является основным компонентом паравиртуализации устройства.

В работе использовалась реализация тулгейта, работающая поверх OTG (open toolgate) - механизма передачи на хост отдельных страниц памяти с данными. OTG работает по следующему принципу: гостевое приложение заполняет регистры процессора необходимыми данными, в том числе адресом передаваемой страницы памяти, и вызывает команду процессора rdpmc. Гипервизор перехватывает этот вызов и анализирует регистры. В результате вызывается механизм передачи и страница гостевой физической памяти копируется в виртуальную память менеджера виртуальных машин (далее - VMapp).

Необходимо отметить, что передача больших буферов осуществляется постранично. При этом происходит дорогостоящее переключение контекста в режим исполнения гипервизора (root mode) и несколько связанных копирований.

Основной функцией тулгейта является вызов указанной в запросе функции в соответствующей библиотеке на хосте, с передачей ему аргументов и буферов. Библиотека должна быть заранее зарегистрирована для использования. Следует отметить важные детали: вызов происходит из гостя к хосту, обрабатывается, и результаты возвращаются в обратную сторону. При этом вызов является синхронным и блокирующим в рамках ядра процессора виртуальной машины. К сожалению, режим асинхронной передачи данных в тулгейте не реализован.

2.1.3. Работа с камерой на хосте

На хосте механизм паравиртуализации камеры реализован в виде библиотеки libhostcamera, которая подключается к VMapp для получения запросов из тулгейта. Для этого в ней реализован требуемый тулгейтом интерфейс вызова функций.

Работа с устройством осуществляется этой библиотекой через стандартный фреймворк OS X, называемый AVFoundation. Этот фреймворк предоставляет Objective-C интерфейс для высокоуровневой работы с видеокамерами, которые могут быть как реальным

аппаратным обеспечением, имеющим специальные драйвера, так и эмулироваться в системе сторонним ПО.

Помимо потокового захвата видеокадров, AVFoundation позволяет осуществлять все основные настройки камеры, такие как количество кадров в секунду (fps), разрешение видео и формат упаковки отдельных пикселей.

Приведём API библиотеки libhostcamera, используемый гостевой библиотекой:

```
open_camera(int num);
close_camera();
start_streaming();
stop_streaming();
get_frame(void *buf);
get_parameters(video_settings_t *settings, int num);
set_parameters(video_settings_t *settings);
```

Основной цикл работы с камерой выглядит следующим образом:

1. open camera - подключение к хостовой камере, регистрация обратных вызовов.
2. parameters - получение поддерживаемых устройством настроек, и установка требуемых гостевым приложением.
3. start streaming - запуск потокового захвата кадров. При получении и обработке каждого последующего кадра фреймворк вызывает обратный вызов CaptureFrame. В нем происходит освобождение предыдущего полученного кадра и захват нового. Захваченное состояние исключает освобождение или повтороное использование этого буфера фреймворком.
4. get frame - передача гостю текущего захваченного кадра.
5. stop streaming - остановка потокового захвата кадров. Освобождение последнего кадра.
6. close camera - отключение от камеры, освобождение вспомогательных объектов.

Библиотека libhostcamera была реализована в рамках данной работы с использованием документации по AVFoundation.

2.2. Требования к производительности

Основные требования к производительности камеры обусловлены необходимостью потоковой передачи данных с заданной частотой (fps). Снижение частоты обновления кадров значительно ухудшает воспринимаемое пользователем качество изображения. Чаще всего используется значение 30 fps. Это обусловлено приемлемым качеством получаемого видеопотока и отсутствием видимых задержек кадров. В некоторых случаях используется 25 fps для снижения объёмов данных, а так же 60 fps и выше в камерах замедленной съёмки. В дальнейшем будем использовать 30 fps там, где это возможно.

Вторым важным фактором, влияющим на качество видеопотока, является разрешение передаваемых кадров. Будем считать минимальным качественным разрешением 640x480 (VGA). Видео высокого качества должно обладать разрешением 1920x1080 (FullHD).

В зависимости от используемого формата упаковки кадров, каждый пиксель изображения может занимать от двух до трёх байтов. Примером стандартной трёхбайтовой упаковки является формат RGB. В этом формате каждый пиксель содержит информацию о красном, зелёном и синем цвете, по байту на каждый. Однако в стеке камеры Android, а так же во многих реальных камерах используется формат упаковки кадров YUYV, основанный на цветовой модели YUV. В этом формате каждая пара пикселей по горизонтали кодируется 4 байтами: по одному байту яркости для обоих пикселей, и два общих байта для цветовых составляющих U и V. В данной работе будет применяться именно YUYV, иначе называемая YUV 422. Таким образом, каждый пиксель изображения кодируется двумя байтами.

Дополнительно рассматривался вопрос целесообразности сжатия передаваемых кадров для уменьшения необходимой пропускной способности. Однако накладные расходы на эффективную упаковку и распаковку оказываются значительно выше выигрыша в меньшем размере передаваемых данных. Вследствие этого было решено отказаться от этой идеи. Данные передаются в "сыром"виде.

Получаем необходимую пропускную способность: $bandwidth = fps * res * 2$. Для VGA разрешения она составляет 18 Мб/с, а для FullHD - 120 Мб/с.

Нельзя забывать, что обработка, запись и вывод кадра на гостевой системе так же занимает некоторое время. Поэтому пропускная способность тулгейта должна быть выше рассчитанной на некоторую величину, определяемую экспериментально.

Аналогичной метрикой производительности является время, затраченное гостевой библиотекой на запрос, обработку и вывод кадра на экран. Это время должно составлять не более $1000/fps$ миллисекунд. В случае задержки кадры могут быть потеряны, что повлечёт за собой снижение качества.

2.3. Тестирование и анализ результатов

2.3.1. Тестирование

Для тестирования производительности паравиртуализованной камеры проводились измерения временных показателей различных операций. Для этого хост–библиотека и библиотека слоя абстракции оборудования на госте подверглись незначительным изменениям. Был внедрен опрос системных таймеров и запись в лог значений затраченного времени.

Главным образом измерялось время передачи данных через тулгейт. Так же производились замеры времени, необходимого на обработку кадров на госте, включающее все остальные операции: перекодировку, отправку кадров приложению, работа с очередью экранных буферов и их заполнение.

Так же, для чистоты эксперимента было измерено время, затрачиваемое хост–библиотекой на обработку запроса и возврат кадра.

Следует отдельно отметить корректность выбранной методики. Данный способ измерения времени не влияет на скорость работы всего механизма и не вносит заметные задержки. Это было проверено с помощью измерения времени, затрачиваемого на единичный опрос таймера. Это время составило значительно меньше 10 микросекунд, что пренебрежимо мало по сравнению с измеряемыми временными интервалами.

Дополнительно проводился анализ лог-файлов на предмет поиска потерянных кадров, а так же уменьшения частоты кадров ниже требуемой.

Определим основное ПО, использованное при тестировании:

- OS X 10.11 в качестве операционной системы хоста;
- Android 5.1 гостевая ОС;
- Parallels Desktop в качестве платформы виртуализации и тулгейта.

Хост системой в данном тесте выступал компьютер Mac Pro 2008 года. Основные характеристики:

- CPU: 2x Intel Xeon 4 cores 3.2 Ghz
- RAM: 16 GB DDR2, 800 Mhz
- Технологии аппаратной виртуализации: VT-x, EPT

Таким образом, в тесте использовалась аппаратная виртуализация процессора и памяти, что привело к высокой скорости исполнения гостя в целом.

В качестве источника кадров на хосте использовались две камеры:

- Реальная аппаратная видеокамера, подключённая по USB. Использовались стандартные драйвера.
- Камера, эмулированная с помощью приложения ManyCam. В качестве картинки использовался видеоролик.

Доступ к обеим камерам осуществлялся совершенно идентично через фреймворк AVFoundation на хосте, как было указано выше. Использование разных камер на хосте никак не повлияло на гостевую часть механизма. Это показало независимость выбранного способа виртуализации от аппаратной составляющей.

2.3.2. Результаты

В результате измерений было получено время передачи через тулгейт отдельных кадров изображения. В зависимости от размера буфера был построен график.

Легко видеть, что зависимость линейная и точки хорошо ложатся на прямую. Отметим, что для каждой точки было собрано по несколько сотен значений, что обеспечило хорошую точность, несмотря на большой разброс отдельных значений.

Полученные результаты: Скорость передачи данных 60 ± 3 Мб/с, время передачи одной страницы данных (4кб) 65 ± 3 мкс.

Благодаря асинхронной обработке полученных от фреймворка кадров, библиотека на хосте обрабатывала запрос на отправку кадра через тулгейт в среднем за 200 мкс. Это время было постоянным для обеих выбранных камер.

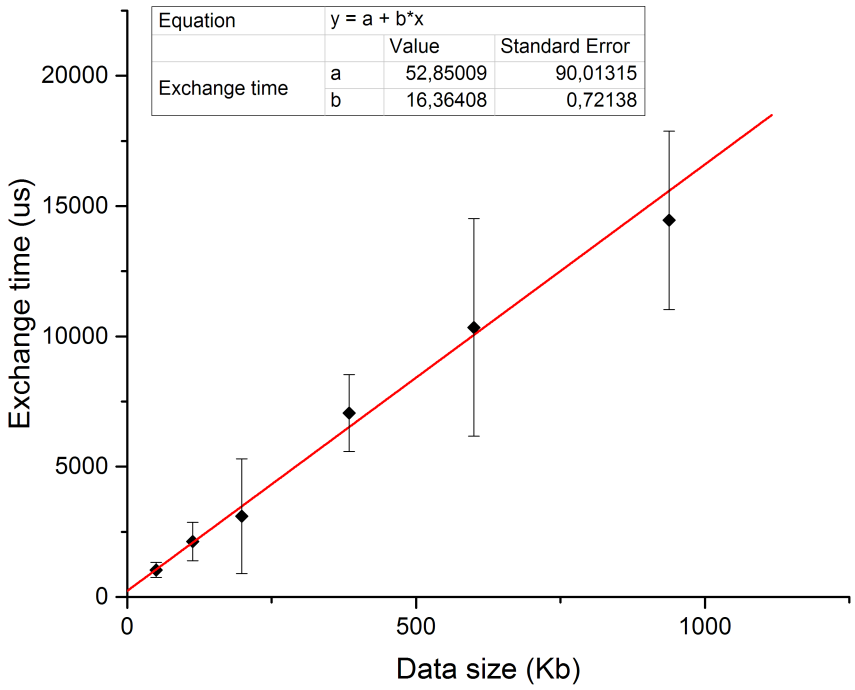

Рис. 4. Зависимость времени передачи буфера от его размера

Полученной производительности паравиртуализованной камеры достаточно для передачи видео разрешением вплоть до 1024x768 (XGA). Для передачи кадра такого разрешения тулгейту необходимо в среднем 25 мс. Оставшегося времени вполне достаточно на обработку кадра гостевой библиотекой. Напомним, в данной работе считается стандартной частота 30 кадров в секунду. В таком случае кадры должны поступать каждые 33 мс.

Несмотря на то, что пропускной способности тулгейта хватает на передачу кадров разрешением 1280x720 (720p), гостевой библиотеке недостаточно времени для обработки изображения, и частота кадров падает ниже необходимой.

Заметим, что производительность тулгейта в целом аналогична производительности протокола USB 2.0, который часто используется для внешних веб-камер.

Итоговая производительность механизма вполне достаточна для комфортного использования пользователем. Недостатком является высокая нагрузка на ядро процессора, на котором осуществляется передача и обработка изображения. Это ведёт к повышенному энергопотреблению, что может быть критично для устройств с батареями.

Проанализируем, насколько полученный механизм паравиртуализации соответствует поставленным целям:

- Доступ - обеспечен доступ к стандартным функциям камеры и возможность настройки её параметров;
- Скорость - скорости работы камеры достаточно для функционирования устройства и передачи видео среднего качества без задержек;
- Мультиплексирование - ввиду специфики устройства камеры, её использование сразу несколькими приложениями невозможно даже в рамках одной операционной системы;
- Изоляция - при любом использовании камера захватывается единственным приложением, что исключает любое постороннее вмешательство;
- Мобильность - как было показано на примере двух камер, зависимости от аппаратного обеспечения камеры не обнаружено.

Одним из способов повышения производительности камеры является использование разных потоков исполнения для передачи и обработки кадров. Однако данный механизм может работать только при выделении гостю более одного виртуального процессора.

Наиболее критичным компонентом оказался тулгейт, скорость которого на порядок ниже скорости копирования данных в оперативной памяти (которая составляет 800 Мб/с в один поток для компьютера на тесте). Расширенная реализация тулгейта позволит в таком же режиме работы добиться вдвое большей скорости передачи данных (120 Мб/с). К сожалению, к моменту проведения измерений расширенная реализация не была доступна для использования. Однако окончательным решением проблем с производительностью передачи данных мог бы быть асинхронный режим работы тулгейта, в котором не происходит блокирования гостевого виртуального процессора.

3. Заключение

В результате проделанной работы был реализован механизм паравиртуализации камеры. Проведённые тесты показали, что производительность виртуализованной камеры оказалась достаточной для передачи видео среднего качества, что подходит для комфортного использования. Однако для видео высокого качества и некоторых других устройств такой производительности может оказаться недостаточно, что оставляет открытым вопрос усовершенстования механизма. Основной проблемой оказалась скорость использованного канала передачи данных между хостом и гостем.

Важным преимуществом использованного механизма является полная независимость от аппаратной составляющей камеры и относительная простота реализации. Основной недостаток — зависимость от гостевой ОС.

3.1. Дальнейшие планы

В дальнейшей работе планируется добавить новые механизмы передачи данных через тулгейт, в частности - асинхронный режим работы, а так же, возможно, режим обратных вызовов (аналог прерываний). Такие режимы работы могут быть необходимы для паравиртуализации более сложных устройств.

Так же важным аспектом будет увеличение скорости передачи данных через тулгейт в целом и паравиртуализация других устройств.

Список литературы

1. Sean Campbell, Michael Jeronimo. *An Introduction to Virtualization.* Intel, 2006.
2. Keith Adams, Ole Agesen. *A Comparison of Software and Hardware Techniques for x86 Virtualization.* ASPLOS, 2006.
3. *Intel Virtualization Technology for Directed I/O. Architecture Specification.* Intel, 2014.
4. Mendel Rosenblum, Carl Waldspurger. *I/O Virtualization.* ACM Queue, 2011.
5. AVFoundation Programming Guide by Apple. `https://developer.apple.com/library/mac/documentation/AudioVideo/Conceptual/AVFoundationPG`
6. Android's camera Hardware Abstraction Layer. `https://source.android.com/devices/camera/index.html`