СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель проведения практики:

Осуществление профессионально-практической подготовки студентов к будущей профессиональной деятельности, закрепление и углубление теоретических знаний в области проектирования информационных систем, овладение навыками воспринимать профессиональные знания и применять их для решения нестандартных задач, в том числе в новой или незнакомой среде при выполнении выпускной квалификационной работы магистра.

Задачи проведения практики:

Выполнить оформление полученных результатов научных исследований по теме ВКР «Метод программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM» в виде разделов РПЗ выпускной квалификационной работы магистра:

— Аннотация ВКР.
— Оглавление ВКР.
— Конструкторский раздел ВКР.
— Исследовательский раздел ВКР.
— Список использованных источников в ВКР.

- Вид практики – производственная.
- Способы проведения практики – стационарная (МГТУ им. Н.Э. Баумана)
- Форма проведения – распределенная – проходит в течение семестра.
- Тип практики – преддипломная практика.

1 Реферат ВКР

Расчетно-пояснительная записка к выпускной квалификационной работе «Метод программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM» содержит 32 страниц, 4 раздела, 15 рисунков, 2 таблиц и 37 используемых источников.

Ключевые слова: доверенная среда исполнения, виртуализация, безопасность, ARM, TrustZone, операционные системы, системное программирование.

Объект разработки: метод программной реализации доверенной среды исполнения

Цель работы: разработка метода программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM.

В аналитическом разделе представлен обзор реализаций доверенных сред исполнения (ДСИ) и описаны их особенности. Сформулированы критерии оценки и проведено сравнение на их основе. Представлена формализованная постановка задачи на разработку метода программной реализации ДСИ с помощью виртуализации процессоров архитектуры ARM.

В конструкторском разделе разработан метод программной реализации ДСИ с помощью виртуализации процессоров архитектуры ARM и представлено его формальное описание в виде диаграмм IDEF0 и схем алгоритмов.

В технологическом разделе обоснован выбор средства программной реализации ДСИ с помощью виртуализации процессоров архитектуры ARM. Разработано ПО, реализующее спроектированный ранее метод.

В исследовательском разделе проведено исследование эффективности и применимости разработанного ПО. Выполнено сравнение результатов работы разработанного метода и метода с аппаратной поддержкой ДСИ на базе процессоров с архитектурой ARM (ARM TrustZone).

Разработанный метод может найти применение в области серверных и облачных технологий.

2 Оглавление ВКР

СОДЕРЖАНИЕ

3 Конструкторский раздел ВКР

В конструкторском разделе разработан метод программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM и представлено его формализированное описание в виде диаграмм IDEF0 и схем алгоритмов. Выполнено проектирование ПО для реализации данного метода.

3.1 Разработка метода программной реализации доверенной среды исполнения

Разработанный метод предполагает использование принципа разделения функциональности на разные компоненты системы. Система спроектирована таким образом, что при атаке или получении доступа к одному из компонентов, её целостность нарушена не будет. Для корректной работы метода, система обязательно должна поддерживать технологию ARM TrustZone и аппаратные механизмы виртуализации ARM.

Каждой гостевой виртуальной машине сопоставляется виртуальная машина выполняющая роль доверенной среды исполнения, для достижения данной цели используется гипервизор. Каждая из этих виртуальных машин выполняется в обычном мире.

Для обеспечения целостности и проверки последовательности загрузки используется безопасный мир (который является частью ARM TrustZone). В безопасном мире располагаются модули отображения адресов памяти гипервизора и виртуальных машин; модуль перехода и сохранения контекста между виртуальными машинами. Такой подход обеспечивает целостность данных, даже если код гипервизора был скомпрометирован.

Каждая ДСИ использует своё индивидуальное рабочее окружение, в котором хранятся данные. Окружение закреплено за конкретной виртуальной машиной и доступное только когда процессор выполняется в режиме hypervisor, с помощью чего и добивается изоляция. При этом, само окружение не является

частью сущности гипервизора.

Для корректного и безопасного взаимодействия вышеописанных компонентов: модулей расположенных в безопасном мире, рабочих окружений и виртуальных машин используется модуль блокировки потока управления, который является частью гипервизора.

На рисунке 1 представлена диаграмма компонентов системы для разработанного метода программной реализации ДСИ.

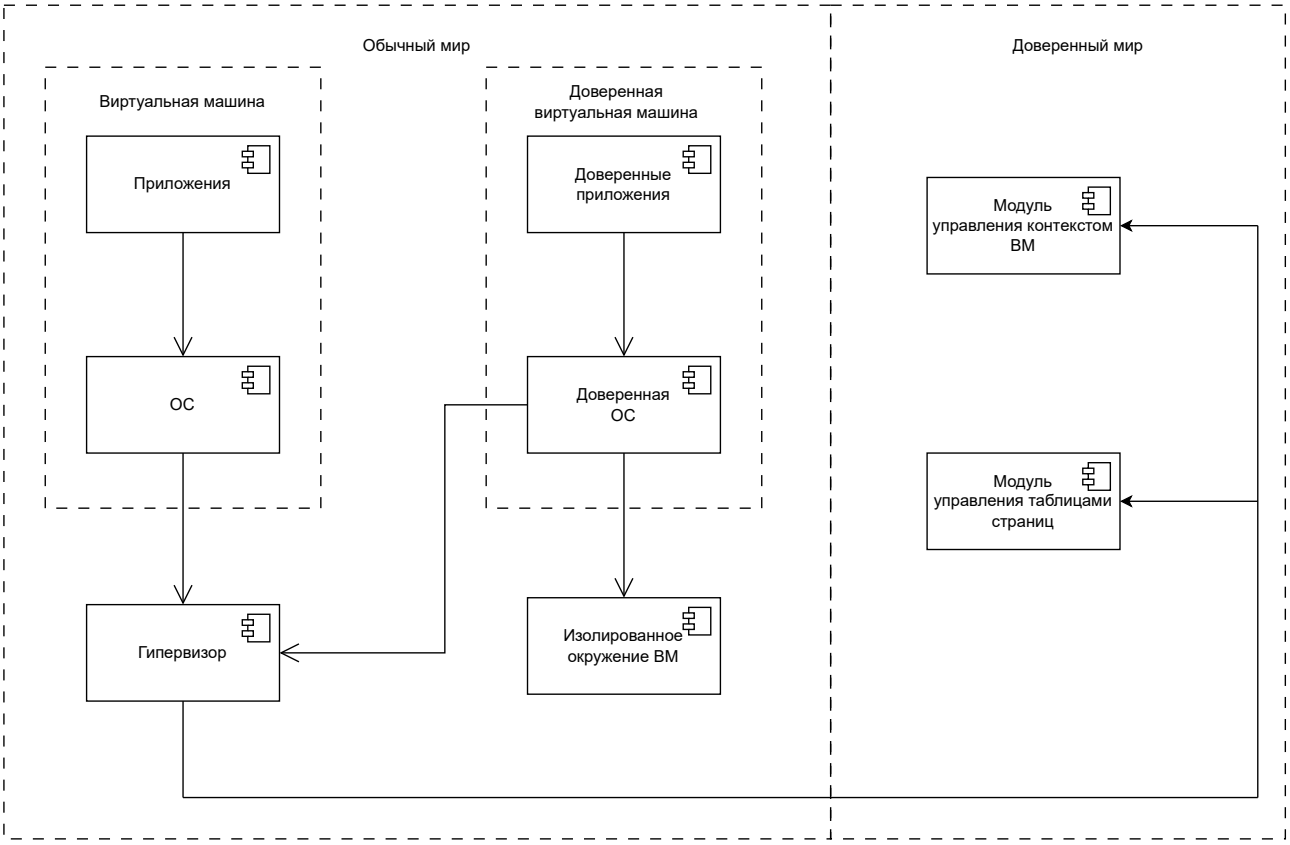

Рисунок 1 – Диаграмма компонентов системы

Далее будет дано детальное описание разрабатываемых модулей и окружений.

3.1.1 Модуль защищенного отображения памяти

Модуль отображения памяти отвечает за трансляцию виртуальных адресов в физические в режиме выполнения hypervisor, а так же промежуточных виртуальных в физические для гостевых виртуальных машин. Модуль выполняется в безопасном мире и предоставляет два интерфейса для гипервизора,

которые позволяют загрузить и модифицировать таблицу страниц.

Система построена таким образом, что модуль отображения памяти обладает монопольным доступом к загрузке и модификации таблицы страниц. Это достигается с помощью допущений, описанных ниже.

— Гипервизор не может изменять регистр, в котором хранится указатель на таблицу страниц: все инструкции, позволяющие это сделать, удаляются из его исходного кода. Сами страницы памяти, на которых располагаются таблицы страниц, помечаются как только для чтения для гипервизора.
— Страницы памяти, на которых расположен код гипервизора, помечаются как только для чтения. Это позволяет гарантировать что исходный код не может быть изменен во время исполнения.
— После запуска системы ни одна страница в адресном пространстве гипервизора не может быть помечена как исполняемая.

Таким образом, для внесения изменения в таблицу страниц необходимо использовать интерфейсы предоставляемые модулем отображения памяти, который управляет и реализует различные политики безопасности на каждое такое изменение.

3.1.2 Модуль блокировки потока управления

Чтобы принудительно передать управление на определенный код при возникновении какого-либо исключения используется модуль блокировки потока управления, который является частью гипервизора.

В архитектуре ARM, при возникновении исключения, управление передаётся специальному обработчику, адрес которого находится в таблице исключений. Код гипервизора модифицирован таким образом, что он лишён возможности модифицировать регистр содержащий базовый адрес таблицы, а так же её элементы. В обработчики исключений, адреса которых указаны в этой таблице, добавлены специальные инструкции, которые гарантируют перенаправление потока управления к необходимым модулям (например, к модулю переключения контекста).

3.1.3 Модуль переключения контекста

Для переключения между контекстами виртуальных машин и гипервизора используется модуль, который выполняется в безопасном мире. Модуль отвечает за переключение между гостевой и доверенной виртуальной машиной и за переключения между виртуальными машинами и гипервизором. Оба типа переключений обрабатываются единообразно, так как в первом случае переключения так же обрабатывает гипервизор.

В архитектуре ARM существует две ситуации, которые могут привести к переключению выполнения из виртуальной машины в гипервизор:

1) аппаратное прерывание;
2) программное прерывание (вызов специальной инструкции процессора) или обработка исключительной ситуации.

В обоих случаях, переключение вызвано исключением, обработка которого будет произведена в данном модуле. Это гарантируется модулем блокировки потока управления.

Гипервизор может переключиться в контекст исполнения виртуальной машины изменив режим привилегий процессора из hypervisor (EL2) в kernel (EL1). Этого можно добиться тремя способами:

1) с помощью инструкции eret;
2) с помощью инструкции movs pc, lr;
3) явно установить режим привилегий.

Все данные вызовы в исходном коде гипервизора должны быть удалены и заменены на соответствующие вызовы предоставляемые модулем переключения контекста.

3.1.4 Индивидуальное рабочее окружение

За каждой доверенной виртуальной машиной закреплено индивидуальное рабочее окружение, выполняемое в режиме работы процессора hypervisor, которое эмулирует функциональность ARM TrustZone. Оно обладает своей таб-

лице страниц, стеком, данными и secure

Каждое окружение удовлетворяет следующим требованиям, которые позволяют его защитить, в случае если гипервизор был скомпрометирован:

— Единая точка входа.
— Запрещены прерывания. Код должен выполняться от точки входа до точки выхода.
— Нет зависимости от данных гипервизора.
— Данные окружения не передаются гипервизору.

Код каждого рабочего окружения загружается по фиксированному адресу в памяти, который задаётся на стадии компиляции, во время инициализации виртуальной машины. Модули, расположенные в безопасном мире, обладают информацией о метаданных каждого рабочего окружения (адрес точки входа и точки выхода). Сами страницы кода окружения помечаются как только для чтения. Первая и последняя выполняемая инструкция – это smc.

Входные данные (получаемые от виртуальной машины), доступны в режиме только для чтения. Для выходных данных выделяются отдельные страницы, так же доступные только для чтения. Чтобы записать в эти страницы какие-либо данные, необходимо сделать запрос к модулю отображения памяти.

Перед тем как передать управление коду рабочего окружения, страницы его стека настраиваются таким образом, что они доступны для чтения и записи только для ядра процессора, на котором сейчас выполняется его код. Так же, страницы его исходного кода помечаются как доступные для выполнения. Противоположные действия выполняются после исполнения последней инструкции рабочего окружения (smc). За эти действия отвечает модуль отображения памяти, находящийся в безопасном мире.

На рисунке 2 представлена диаграмма потоков данных для операции записи в хранилище рабочего окружения в нотации Йордона-Де Марко.

Рисунок 2 – Диаграмма потоков данных операции записи в хранилище рабочего окружения

3.2 Формализированное описание метода

Разрабатываемый метод – это виртуализация механизмов защиты, которые предоставляет аппаратная реализация ДСИ ARM TrustZone. На рисунках 3 - 4 представлена IDEF0-диаграмма и её детализированная версия метода программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM. В следующих разделах будет представлено более подробное описание механизмов представленных на рисунке 4.

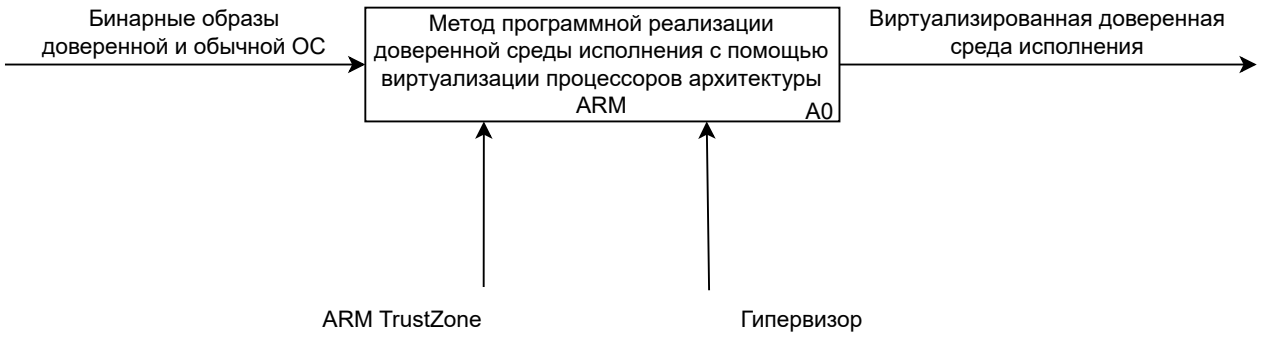

Рисунок 3 – IDEF0-диаграмма разработанного метода

3.2.1 Описание доверенной загрузки

Доверенная загрузка используется для обеспечения целостности загрузки системы. Процесс загрузки устройства с поддержкой ARM TrustZone включает в себя пять этапов.

— Загрузка загрузчика ОС из защищенного от воздействия внешних источников ПЗУ.

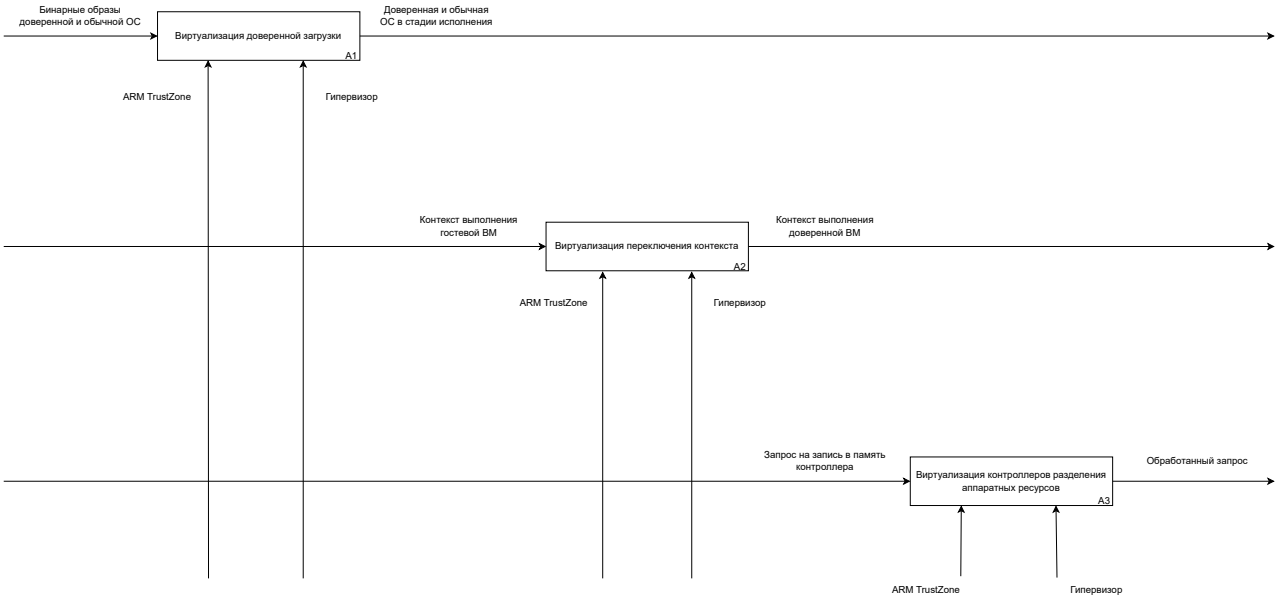

Рисунок 4 – Детализированная IDEF0-диаграмма разработанного метода

— Инициализация окружения и загрузка ядра доверенной ОС.
— Доверенное ядро инициализирует и настраивает окружение для корректной работы безопасного мира.
— Доверенное ядро загружает в память загрузчик обычной ОС и передаёт ему управление.
— Загрузчик обычной ОС вычисляет контрольную сумму бинарного образа ОС перед его загрузкой в память и загружает ОС.

Для виртуализации доверенной загрузки необходимо обеспечить следующие свойства:

1) ядро доверенной ОС должно загрузиться до загрузки обычного;
2) образ обычной ОС должен быть проверен;
3) образ ОС не может быть подменён.

На рисунке 5 представлена детализированная IDEF0-диаграмма виртуализации доверенной загрузки. Данная схема полностью удовлетворяет свойствам, которые были описаны выше.

На рисунке 6 представлена схема алгоритма регистрации виртуальных машин в безопасном мире (в модуле переключения контекста).

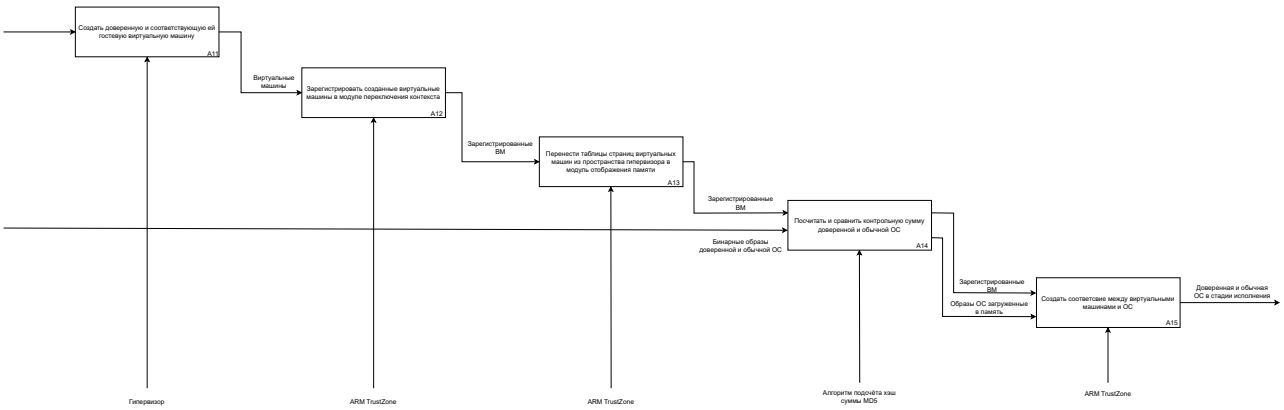

Рисунок 5 – Детализированная IDEF0-диаграмма виртуализации доверенной загрузки

3.2.2 Описание защиты и переключение контекста выполнения

Каждый из миров (безопасный и обычный) имеют свой контекст выполнения – значения регистров, настройка памяти, структуры данных и так далее. Реализация ARM TrustZone предоставляет функциональности защиты этого контекста – каждый контекст доступен для чтения и записи только из мира, к которому он принадлежит.

На рисунке 7 представлена детализированная IDEF0-диаграмма виртуализации переключения контекста между гостевой виртуальной машиной и доверенной. Контекст выполнения остаётся защищенным, т.к. сохраняется и восстанавливается он из безопасного мира. Обратная схема переключения контекста (из доверенной в гостевую ВМ) аналогична.

На рисунке 8 представлена схема алгоритма сохранения контекста выполнения виртуальной машины в безопасном мире.

3.2.3 Описание разделения аппаратных ресурсов

ARM TrustZone разделяет аппаратные ресурсы (память, периферия, прерывания) между безопасным и обычном миром. Для каждого из этих ресурсов существует отдельный контроллер, который отвечает за их настройку и распределение между мирами. Все три контроллера могут быть настроены только из безопасного мира. Таким образом, необходимо виртуализировать каждый их

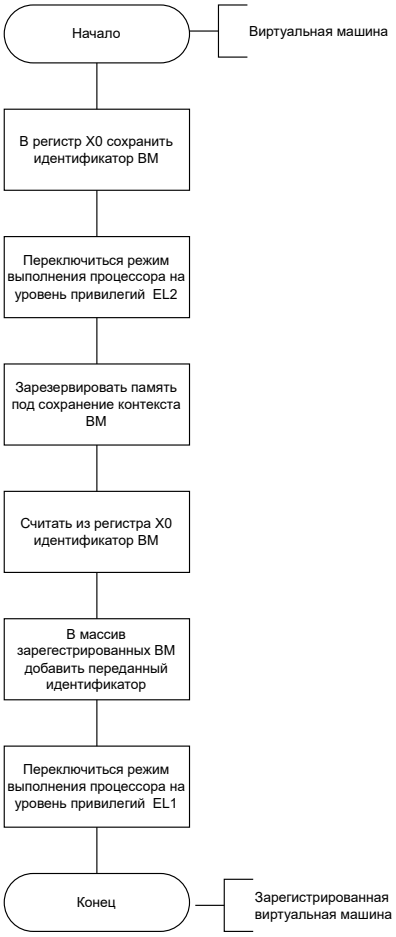

Рисунок 6 – Схема алгоритма регистрации виртуальных машин в безопасном мире

этих контроллеров.

Для каждой виртуальной машины область памяти в которой находятся контроллеры помечена как только для чтения. В результате попытки записи в эти контроллеры, процессор вызывает исключение и управление передаётся гипервизору для его дальнейшей обработки. Такой подход называется trap-and-emulate [23].

На рисунке 9 представлена детализированная IDEF0-диаграмма виртуализации контроллеров разделения аппаратных ресурсов.

Вывод

Был разработан метод программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM. Были спроектированы и разработаны следующие компоненты:

Рисунок 7 – Детализированная IDEF0-диаграмма виртуализации переключения контекста

- модуль защищенного отображения памяти;
- модуль блокировки потока управления;
- модуль переключения контекста;
- индивидуальное рабочее окружение.

Представлено формализированное описание метода в виде IDEF0-диаграммы, состоящей из трех уровней, которые так же были детализированы:

- виртуализация доверенной загрузки;
- виртуализация переключения контекста;
- виртуализация контроллеров разделения аппаратных ресурсов.

С помощью схем алгоритмов описаны используемые в разработанном методе алгоритмы:

- регистрация виртуальных машин в безопасном мире;
- сохранение контекста выполнения виртуальной машины.

Рисунок 8 – Схема алгоритма контекста выполнения виртуальной машины

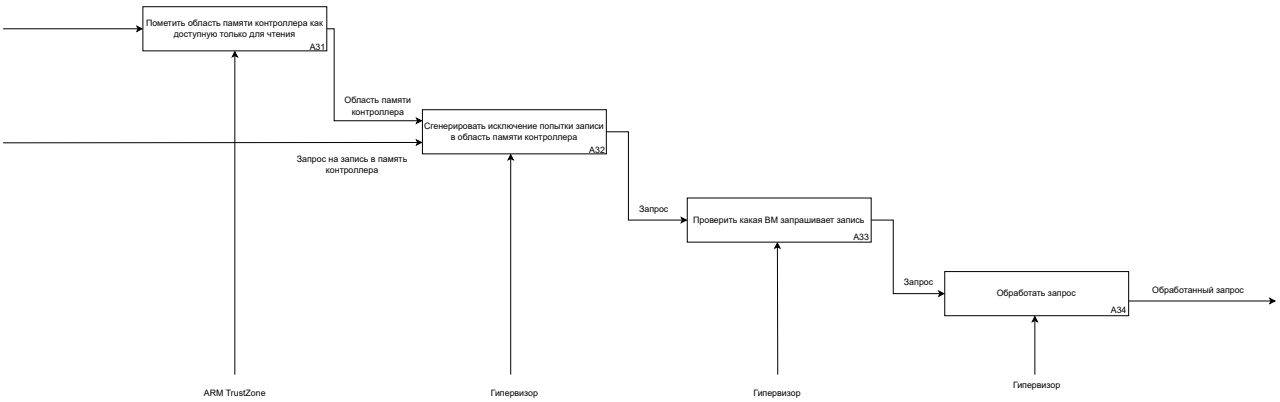

Рисунок 9 – Детализированная IDEF0-диаграмма виртуализации контроллеров разделения аппаратных ресурсов

4 Исследовательский раздел ВКР

В данном разделе проведено исследование эффективности и применимости разработанного программного обеспечения. Выполнено сравнение результатов работы разработанного метода и метода с аппаратной поддержкой доверенной среды исполнения на базе процессоров с архитектурой ARM (ARM TrustZone).

4.1 Методика проведения исследования

Для того чтобы иследовать эффективность и применимость разработанного программного обеспечения, необходимо сравнить количество машинных инструкций для выполнения задач ДСИ: смена контекста между мирами, проверка целостности, обработку прерываний и так далее. Для точного подсчета количества выполняемых инструкций за промежуток времени, был использовано аппаратное расширение ARM Performance Monitoring Unit [25]. Было подсчитано количество инструкций для выполнениях одинаковых задач с использлванием виртуализации ARM TrustZone и аппаратного решения.

В качестве ядра гостевой ОС был использовано ядро Linux версии 6.15, а в качестве ядра привелигированной ОС – OP-TEE версии 4.0. В качестве гипервизора был использован KVM, который является частью ядра Linux.

Сравнение было проведено как для 32-битных процессоров с архитектурой ARMv7 так и для более новых, 64-битных процессоров с архитектурой ARMv8. БЫли выбраны два устройства: Raspberry Pi 4 Model B [26] и Raspberry Pi 2 Model [27]. Raspberry Pi 4 Model B обладает следующими характеристиками:

— Четыре 64-битных ядра Cortex A72 (ARMv8) с тактовой частотой 1,5Ггц.
— 8 Гб ОЗУ.

Raspberry Pi 2 обладает следующими характеристиками:

— Четыре 32-битных ядра Cortex A7 (ARMv7) с тактовой частотой 0.9ГГц.
— 1 Гб ОЗУ.

Во время проведения исследования для каждой виртуальной машины было выделен 1 виртуальный CPU который соотвествует 1 физическому CPU.

4.2 Сравнение количества машинных инструкций с аппаратной реализацией

4.2.1 Сравнение при выполнение ключевых задач

Можно выделить три ключевые задачи, выполняемых доверенной средой исполнения, без которых она не может считаться полноценной:

1) смена контекста выполнения между гостевым и доверенным миром;
2) разделение аппаратных ресурсов;
3) проверка целостности загружаеммых образов ОС.

В таблице 1 представлено сравнение количества машинных инструкций для смены контекста выполнения и разделения аппаратных ресурсов между мирами: разделение памяти и прерываний. В первых двух столбцах указаны результаты сравнения на устройстве Raspberry Pi 2 Model B; первый столбец – аппаратная реализация, второй – виртуализация (разработанный метод). В третьем и чертвёртом столбце указанные результаты сравнения выполняемые на устройстве Raspberry Pi 4 Model, для аппаратной и виртуализированной реализации соотвественно.

Таблица 1 – Сравнение количества инструкций необходимых для выполнения ключевых задач ДСИ

	R Pi2 (A)	**R Pi2 (B)**	**R Pi4 (A)**	**R Pi4 (B)**
Смена контекста	9200	18745	1525	7625
Разделение участков памяти	3245	7055	2208	7800
Разделение прерываний	2273	6251	986	3421

По результатам из таблицы 1 можно сделать вывод, что смена контекста для 32-битных процессоров в разработанном методе требует в два раза больше инструкций, чем в аппаратной реализации ARM TrustZone. Для 64-битных

процессоров разница составляет порядка 5 раз. Данную разницу можно счесть приемлимой, так как смена контекста между мирами происходит редко и практически никак не отражается на производительности выполняемых приложений и всей системы в целом.

Для выполнения разделения участков памяти, по сравнению с аппаратной реализацией, необходимо в 2 и 4 раза для 32-битных и 64-битных процессоров соотвественно. Для разделения и корректной маршрутизации прерываний необходимо в 3 раза больше инструкций как для 32-битных, так и для 64-битных процессоров. Данную разницу, так же, как и в случае со сменой контекста, можно счесть приемлимой, так как операции разделения ресурсов выполняются редко.

В таблице 2 представлено сравнение количества машинных инструкций для проверки целостности загружаемых образов ОС. Для этого было проведено хэширование регионов памяти размером 1, 16, 32, 64 и 128 Кб с помощью алгоритма SHA256 [28].

Таблица 2 – Сравнение количества инструкций необходимых для выпонения проверки целотсности

	R Pi2 (A)	**R Pi2 (B)**	**R Pi4 (A)**	**R Pi4 (B)**
1 Кб	1150	1256	300	325
16 Кб	17890	19200	4450	4800
32 Кб	36502	38600	10200	11223
64 Кб	80215	84555	22254	23507
128 Кб	165865	170205	50700	51998

По результатам из таблицы 2, можно сделать вывод что накладные расходы для выполнения проверки целостности в среднем составляют 5-10% по сравнению с аппаратной реализацией, что не является критичным и не влияет на производительность системы.

4.2.2 Сравнение с использованием пользовательских приложений

Для сравнения накладных ресурсов при использовании пользовательских приложений были выбраны приложения ccrypt и GoHttp. ccrypt был использован для шифрования файлов размеров 1 Кб, а GoHttp для передачи по сети. Логика шифрования данных реализована на уровне доверенной среды исполнения.

Было проведено сравение количества выполняемых как и с одной парой (гостевая и доверенная ОС) виртуальных машин, так и при запуске нескольких пар одновременно.

На рисунках 10 и 11 представлено сравнение (в процентах) количества используемых инструкций при использовании одной пары ВМ. Аппаратная реализция отмечена как 100%.

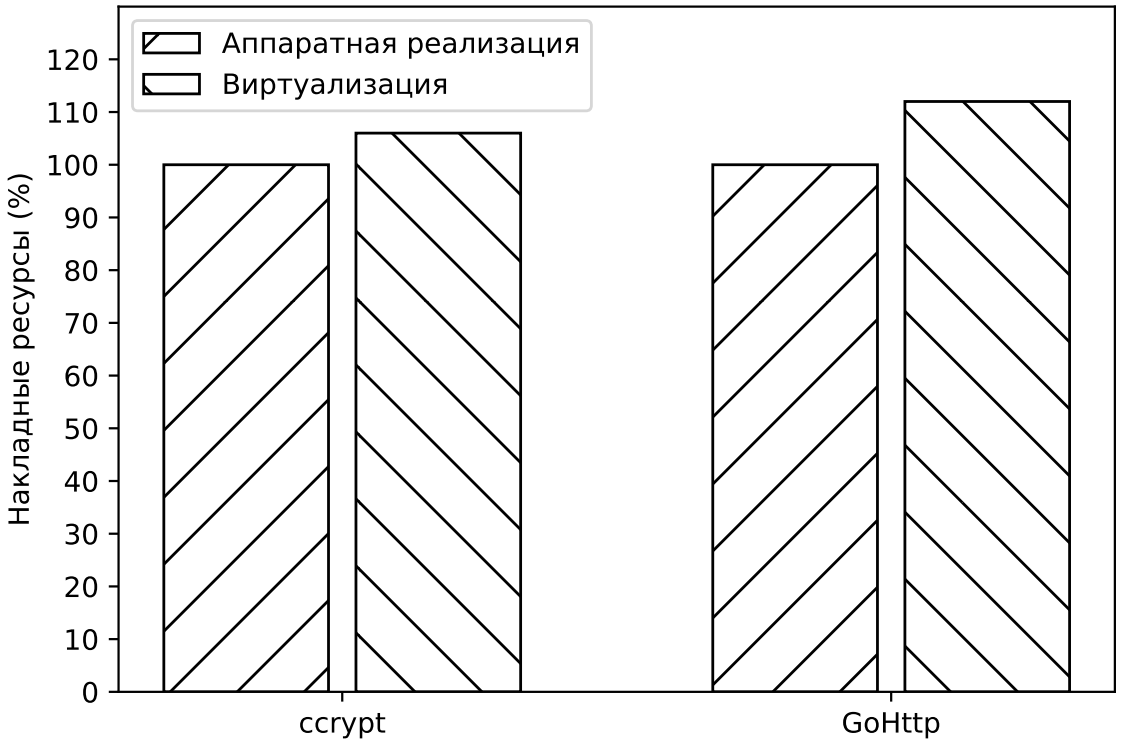

Рисунок 10 – Сравнение результатов выполнения пользовательских приложений (ARMv7)

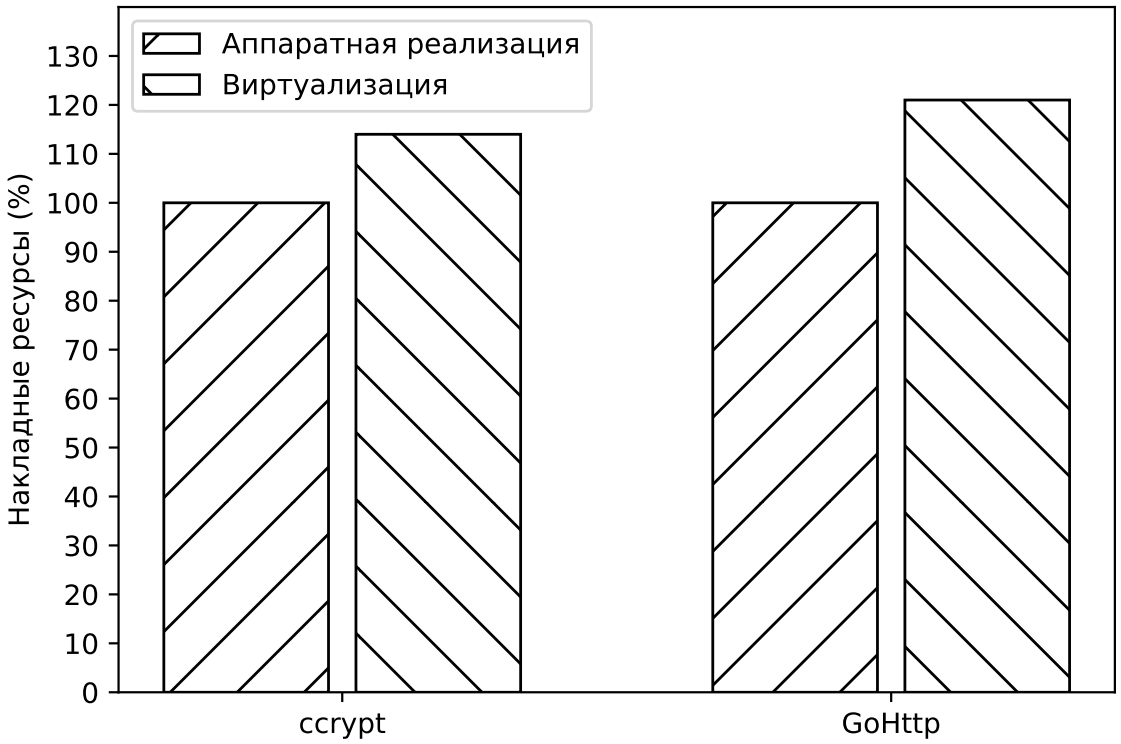

Рисунок 11 – Сравнение результатов выполнения пользовательских приложений (ARMv8)

На рисунках 12 и 13 представлено сравнение количества инструкций при использовании нескольких пар ВМ, которые передают файлы размером 1Кб между друг другом с использованием приложения GoHTTP.

Можно сделать вывод, что при использовании одной пары ВМ, накладные ресурсы на исполнение пользовательских приложений в среднем составляют 7-15%. При использовании нескольких пар ВМ, накладные ресурсы возрастают и составляют от 20 до 50% по сравнению с аппаратной реализацией. Заметный рост накладных ресурсов наблюдается при использовании двух пар ВМ (20% и 35% для ARMv7 и ARMv8 соотвественно), но незначительный рост в 5-10% при увеличении количества пар (две и более). Можно сделать вывод, что ключевую роль в увеличении накладных расходов играет тот факт, что параллельно используется более одной пары ВМ, а не зависит от их количества: разница при

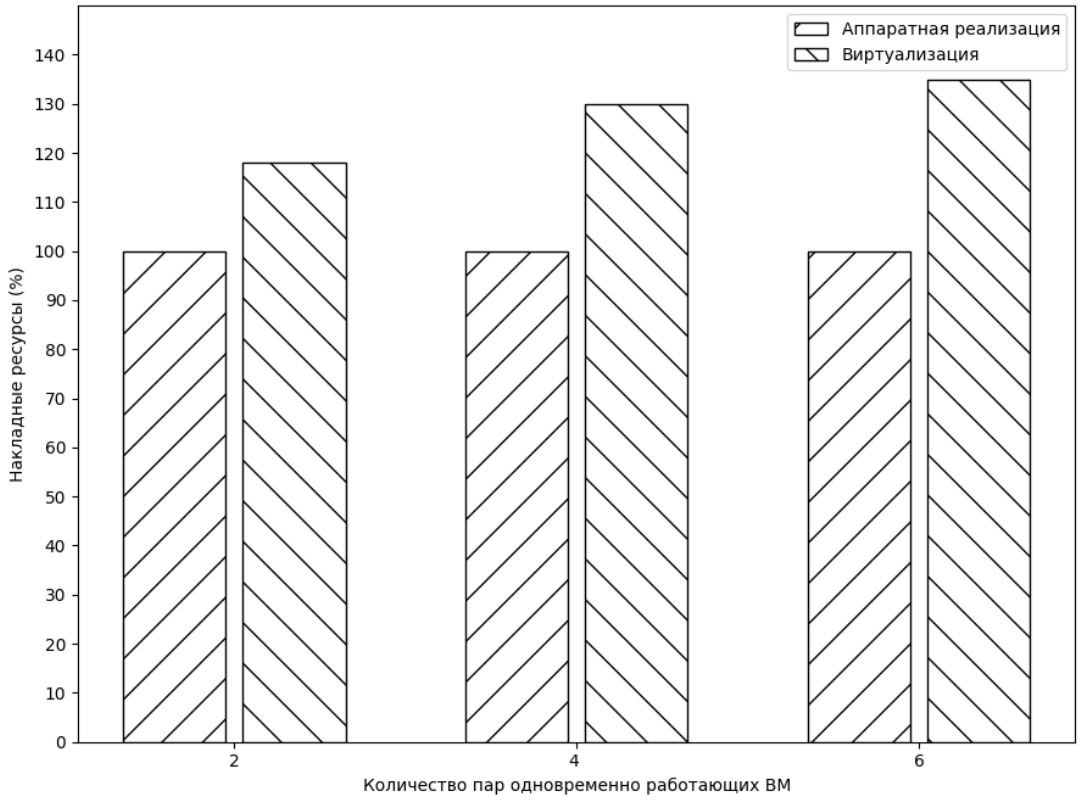

Рисунок 12 – Сравнение результатов выполнения пользовательских приложений (ARMv7), несколько пар ВМ

использовании 2 и 6 пара ВМ составляет 7-15%.

4.2.3 Сравнение с использованием серверных приложений

Для тестирования накладных ресурсов при использовании серверных приложений были выбраны MongoDB [29] и Apache [**?**]. Количество виртуальных CPU для каждой ВМ было увеличено до 4. Используется одна пара ВМ. В качестве устройства использовался Raspberry Pi 4 Model B (ARMv8).

На рисунке 14 представлены результаты сравнения с использованием MongoDB
клиент, расположенный на той же ВМ, что и сервер, на протяжении 1 минуты вставляет объекты различного размера в таблицу базы данных.

Из риснука 14 можно сделать вывод о том, что при использовании разработанного метода скорость записи в сервер MongoDB практически идентична

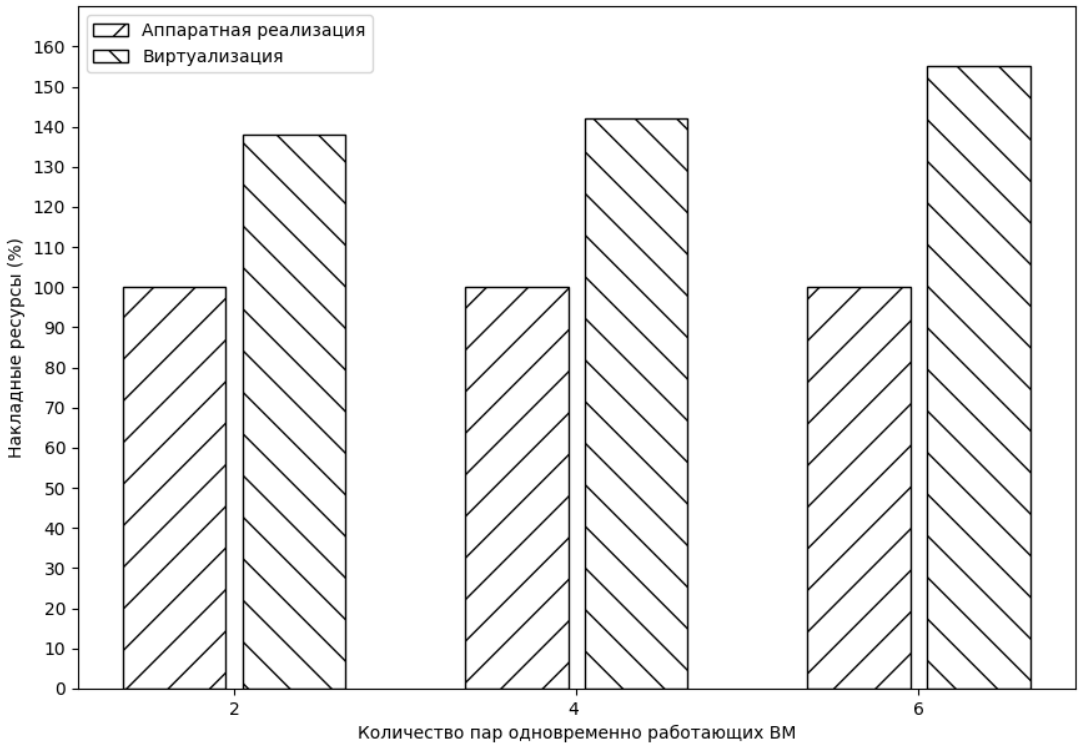

Рисунок 13 – Сравнение результатов выполнения пользовательских приложений (ARMv8), несколько пар ВМ

скорости записи при использовании аппаратного метода: разница соствляет 5-10%.

На рисунке 15 представлены результаты сравнения с использованием Apache: клиент, расположенный на той же ВМ, что и сервер, на протяжении 1 минуты скачивает файл различного размера с сервера.

Из риснука 14 можно так же сделать вывод, что скорость чтения с сервера Apache при использовани разработанного метода близка к скорости с использованием аппаратной реализации: разница составляет 10-15%.

Вывод

В данном разделе проведено исследование эффективности разработанного программного обеспечения. Было произведено сравнение количества используемых машинных инструкций для выполнения ключевых задач ДСИ:

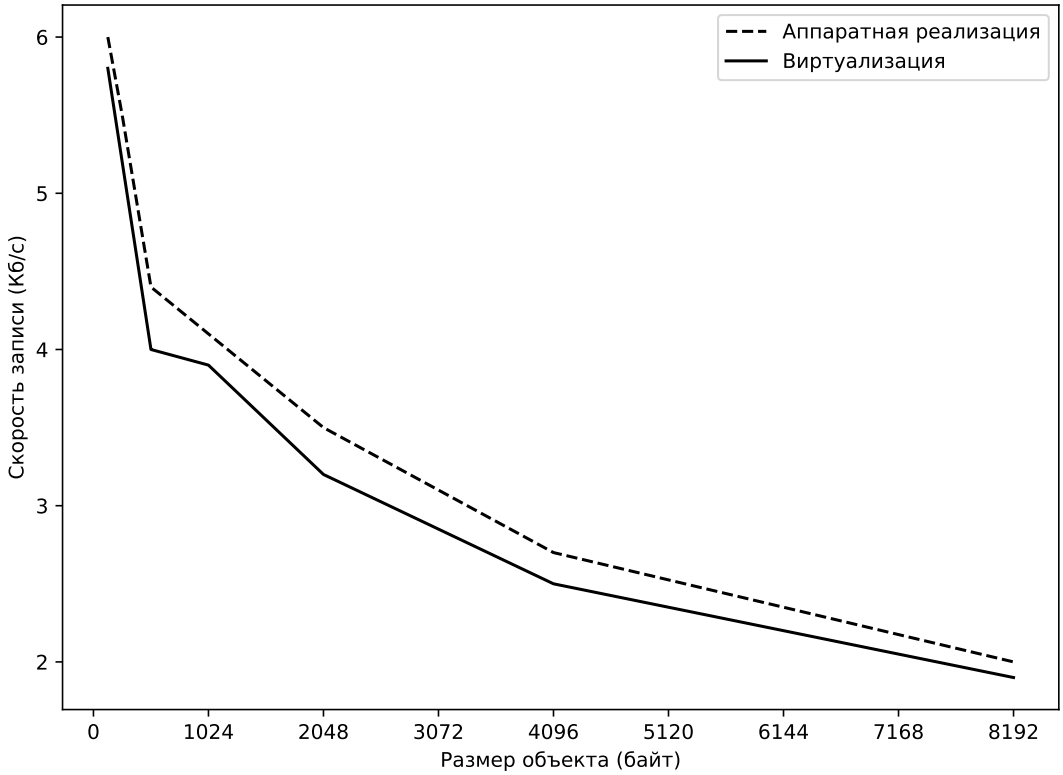

Рисунок 14 – Зависимость скорости записи на сервер MongoDB от размера объекта

— Для смены контекста требуется в 2 и 5 раз больше инструкций для 32-битных (ARMv7) и 64-битных (ARMv8) процессоров соотвественно.
— Для обработки разделения аппаратных ресурсов в среднем требуется в 2-4 раза больше инструкций.
— Разница в количестве инструкций для проверки целостности образов ОС между разработанным методом и аппаратной реализацией в среднем составляет 5-10%.

Во всех приведенных сравнениях, для 32-битных процессоров разница в количестве инструкций составляет меньше, чем для 64-битных (в среднем в 1.5-2 раза).

Было проивзедено сравнение накладных ресурсов при использовании пользовательских и серверных приложений:

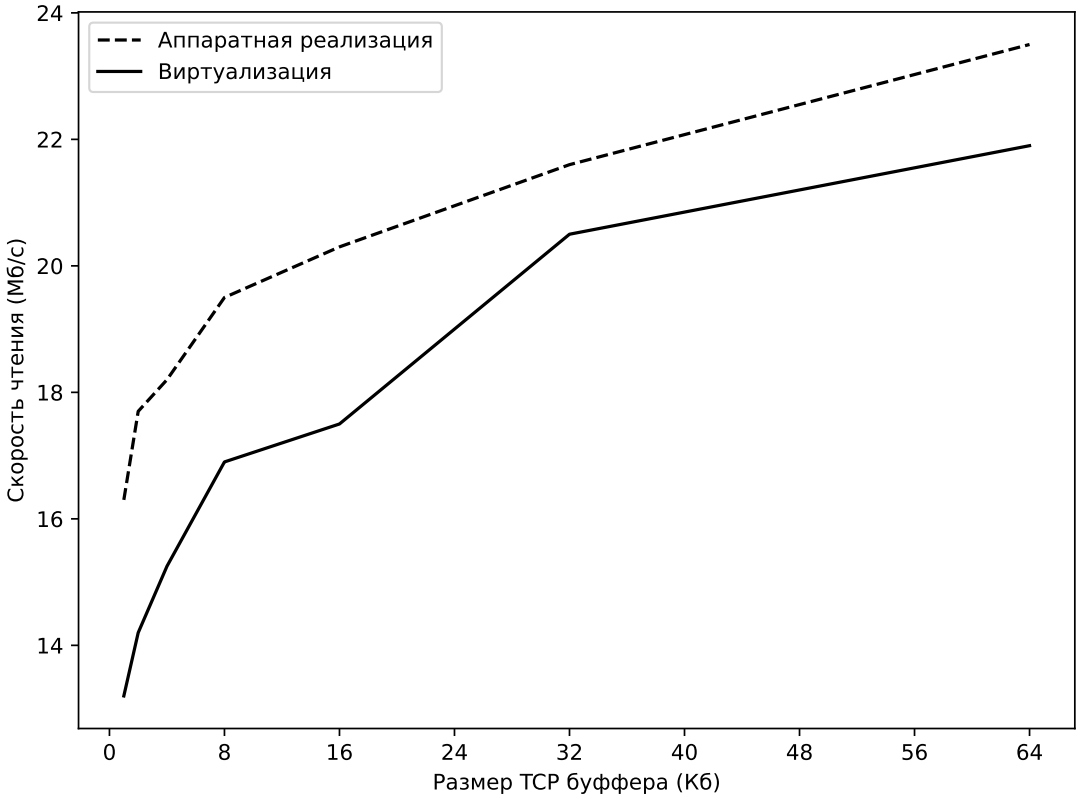

Рисунок 15 – Зависимость скорости чтения данных с сервера Apache от размера TCP буффера

- Для пользовательских приложений разница с аппаратной реализацией составляет 7-15% и 20-50% при использовании несольких виртуальных машин одновременно.
- Накладные расходы резко возрастают (на 20-40%) если в системе используется более одной пары ВМ.
- Произведено сравнение скорости записи данных в сервер MongoDB: разница с аппаратной реализацией составляет 5-10 в зависимости от размера данных%.
- При чтении файлов с сервера Apache было установелно, что скорость чтения на 10-15% меньше, чем при использовании аппаратной реализации.

5 Список использованных источников ВКР

1. Introduction to Trusted Execution Environments – Global Platform [Электронный ресурс]. – Режим доступа: https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf, свободный – (09.10.2023)

2. Intel | Data Center Solutions, IoT, and PC Innovation [Электронный ресурс]. – Режим доступа: https://www.intel.com/, свободный – (10.11.2023)

3. Building the Future of Computing – Arm® [Электронный ресурс]. – Режим доступа: https://www.arm.com, свободный – (09.10.2023)

4. The Security Paradox of Complex Systems, 2002. David Woods, Nancy Leveson, Brian Rebentisch. c. 1 - 15.

5. Comparison of Prominent Trusted Execution Environments, 2022. Xiaoyu Zhang [Электронный ресурс]. – Режим доступа: https://elib.uni-stuttgart.de/bitstream/11682/12171/1/Zhang_Xiaoyu_Prominent_Trusted_Execution_E – (22.10.2023)

6. TrustedFirmware-A (TF-A) | ARM [Электронный ресурс]. – Режим доступа: https://www.trustedfirmware.org/projects/tf-a/ – (22.10.2023)

7. Secure Monitor Calling Convention (TF-A) | ARM [Электронный ресурс]. – Режим доступа: https://developer.arm.com/Architectures/SMCCC – (22.10.2023)

8. OP-TEE | ARM [Электронный ресурс]. – Режим доступа: https://www.trustedfirmware.org/projects/op-tee/ – (22.10.2023)

9. Global Platform – TEE Client API Specification [Электронный ресурс]. – Режим доступа: https://globalplatform.org/wp-content/uploads/2010/07/TEE_Client_API_Specification-V1.0.pdf – (22.10.2023)

10. Global Platform – TEE Internal Core API Specification [Электронный ресурс]. – Режим доступа: https://globalplatform.org/specs-library/tee-internal-core-api-specification/ – (22.10.2023)

11. Diffie-Hellman key agreement | IBM [Электронный ресурс]. – Режим доступа: https://www.ibm.com/docs/en/zos/2.1.0?topic=ssl-diffie-hellman-key-agreement – (27.10.2023)

12. FIPS 180-2, Secure Hash Standard [Электронный ресурс]. – Режим доступа: https://csrc.nist.gov/files/pubs/fips/180-2/final/docs/fips180-2.pdf – (27.10.2023)

13. Attestation Services for Intel® Software Guard Extensions [Электронный ресурс]. – Режим доступа: https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/attestation-services.html – (27.10.2023)

14. Keystone Enclave Documentation [Электронный ресурс]. – Режим доступа: https://buildmedia.readthedocs.org/media/pdf/keystone-enclave/docswork/keystone-enclave.pdf – (01.11.2023)

15. Keystone Basics | Keystone Enclave [Электронный ресурс]. – Режим доступа: http://docs.keystone-enclave.org/en/latest/Getting-Started/How-Keystone-Works/Keystone-Basics.html#overview – (01.11.2023)

16. TS-perf: Performance Measurement of Trusted Execution Environment and Rich Execution Environment on Different CPUs, 2021. Kuniyasu Suzaki Kenta Nakajima Tsukasa Oi Akira Tsukamoto

17. Power Analysis Attack – Revealing the Secrets of Smart Cards. Stefan Mangard, Elisabeth Oswald, Thomas Popp, 2007.

18. HARDWARE ATTACK DETECTION AND PREVENTION FOR CHIP SECURITY | Jaya Doef, University of New

Hampshire. [Электронный ресурс] – Режим доступа: https://scholars.unh.edu/cgi/viewcontent.cgi?article=2027&context=thesis – (01.11.2023)

19. Exploiting the DRAM rowhammer bug to gain kernel privileges [Электронный ресурс] – Режим доступа: https://www.blackhat.com/docs/us-15/materials/us-15-Seaborn-Exploiting-The-DRAM-Rowhammer-Bug-To-Gain-Kernel-Privileges.pdf – (01.11.2023)

20. Linux – Open Source Operating System [Электронный ресурс] – Режим доступа: https://www.linux.org/ – (18.02.2024)

21. KVM – Kernel-based Virtual Machine [Электронный ресурс] – Режим доступа: https://linux-kvm.org/ – (18.02.2024)

22. QEMU – Quick Emulator [Электронный ресурс] – Режим доступа: https://www.qemu.org/ – (18.02.2024)

23. Learn the architecture - AArch64 virtualization [Электронный ресурс] – Режим доступа: https://developer.arm.com/documentation/Trapping-and-emulation-of-instructions – (01.04.2024)

24. vTZ: Virtualizing ARM TrustZone [Электронный ресурс] – Режим доступа: https://ipads.se.sjtu.edu.cn/_media/publications/vtz-security17.pdf (08.04.2024)

25. Arm CoreSight Performance Monitoring Unit Architecture [Электронный ресурс] – Режим доступа: https://developer.arm.com/documentation/ihi0091/latest/ (11.05.2024)

26. Raspberry Pi 4 Model B | Raspberry [Электронный ресурс] – Режим доступа: https://www.raspberrypi.com/products/raspberry-pi-4-model-b (11.05.2024)

27. Raspberry Pi 2 Model B / Raspberry [Электронный ресурс] – Режим доступа: https://www.raspberrypi.com/products/raspberry-pi-2-model-b (11.05.2024)

28. Алгоритм SHA-256 | Google [Электронный ресурс] – Режим доступа: https://support.google.com/google-ads/answer/9004655 (11.05.2024)

29. MongoDB: The Developer Data Platform | MongoDB [Электронный ресурс] – Режим доступа: https://www.mongodb.com/ (11.05.2024)

30. Welcome! - The Apache HTTP Server Project [Электронный ресурс] – Режим доступа: https://httpd.apache.org/ (11.05.2024)

ЗАКЛЮЧЕНИЕ

В ходе преддипломной практики выполнено оформление полученных результатов научных исследований по теме ВКР «Метод программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM»:

— Аннотация ВКР.
— Оглавление ВКР.
— Конструкторский раздел ВКР.
— Исследовательский раздел ВКР.
— Список использованных источников в ВКР.