Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «**Информатика и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАЧУНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Исследование эффективности и применимости метода программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM»

Студент группы **ИУ7-42М** _____ **А. В. Романов**
(Подпись, дата) (И.О. Фамилия)

Руководитель НИР _____ **Д. Е. Бекасов**
(Подпись, дата) (И.О. Фамилия)

***2024** г.*

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Необходимость повышения безопасности исполнения приложений, работающих в системах безопасности и обрабатывающих защищаемую информацию, привела к разработке программно-аппаратных решений, создающих доверенные среды исполнения (англ. TEE – Trusted Execution Environment [1]) на базе аппаратных средств, доверенных загрузок или аппаратно-программных модулей доверенной загрузки. Intel [2] и ARM [3] являются лидерами в этой области. Целью данной работы является исследование эффективности и применимости метода программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести исследование эффективности и применимости ПО;
- выполнить сравнение метода программной реализации с аппаратным методом.

1 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).

1.1 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 [**?**]. 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 [**?**] и Raspberry Pi 2 Model [**?**]. 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 бы-

ло выделен 1 виртуальный CPU который соотвествует 1 физическому CPU.

1.2 Сравнение количества машинных инструкций с аппаратной реализацией

1.2.1 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 результам 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 [**?**].

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.

1.2.2 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 1 и 2 представлено сравнение (в процентах) количества используемых инструкций при использовании одной пары ВМ. Аппаратная реализция отмечена как 100%.

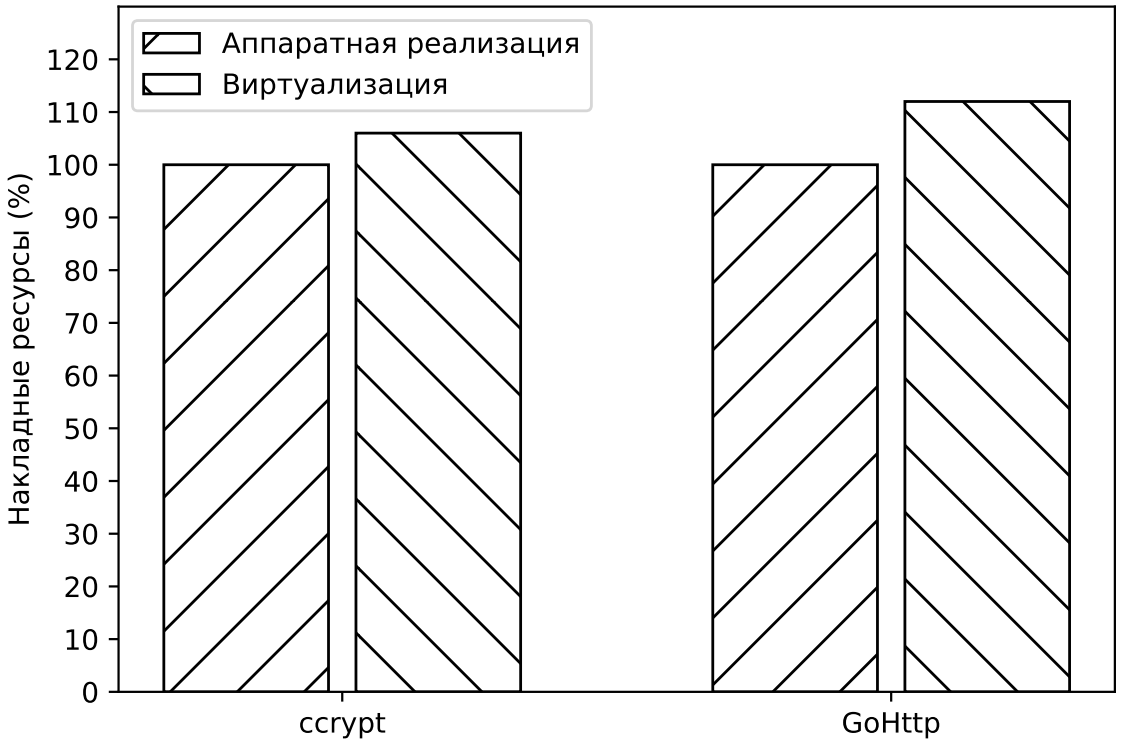

Рисунок 1 – Сравнение результатов выполнения пользовательских приложений (ARMv7)

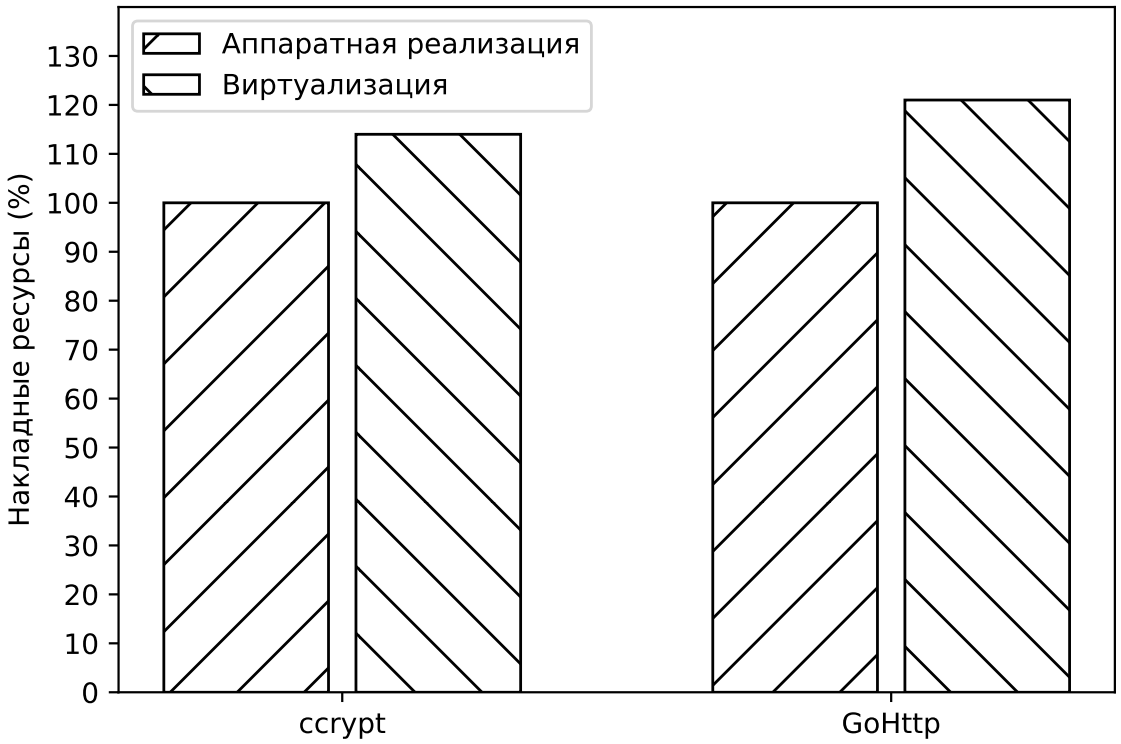

Рисунок 2 – Сравнение результатов выполнения пользовательских приложений (ARMv8)

На рисунках 3 и 4 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

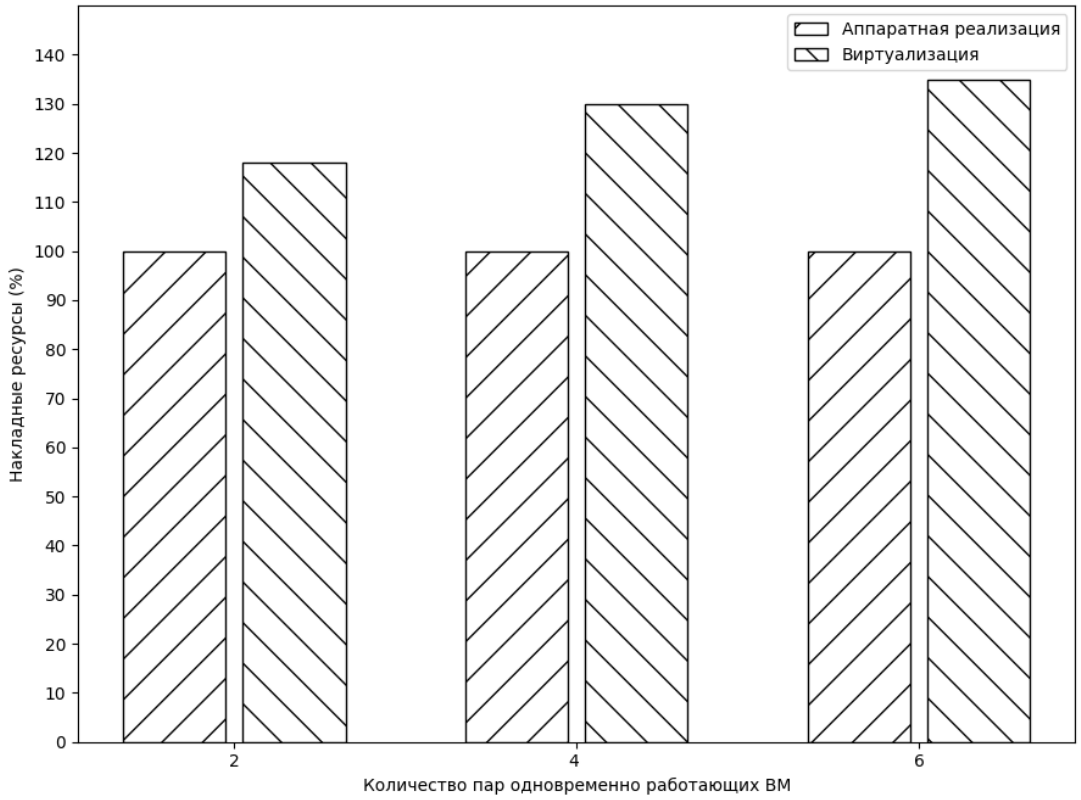

Рисунок 3 – Сравнение результатов выполнения пользовательских приложений (ARMv7), несколько пар ВМ

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

1.2.3 Сравнение с использованием серверных приложений

Для тестирования накладных ресурсов при использовании серверных приложений были выбраны MongoDB [**?**] 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 5 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 5 можно сделать вывод о том, что при использовании разработанного метода скорость записи в сервер MongoDB практически идентична

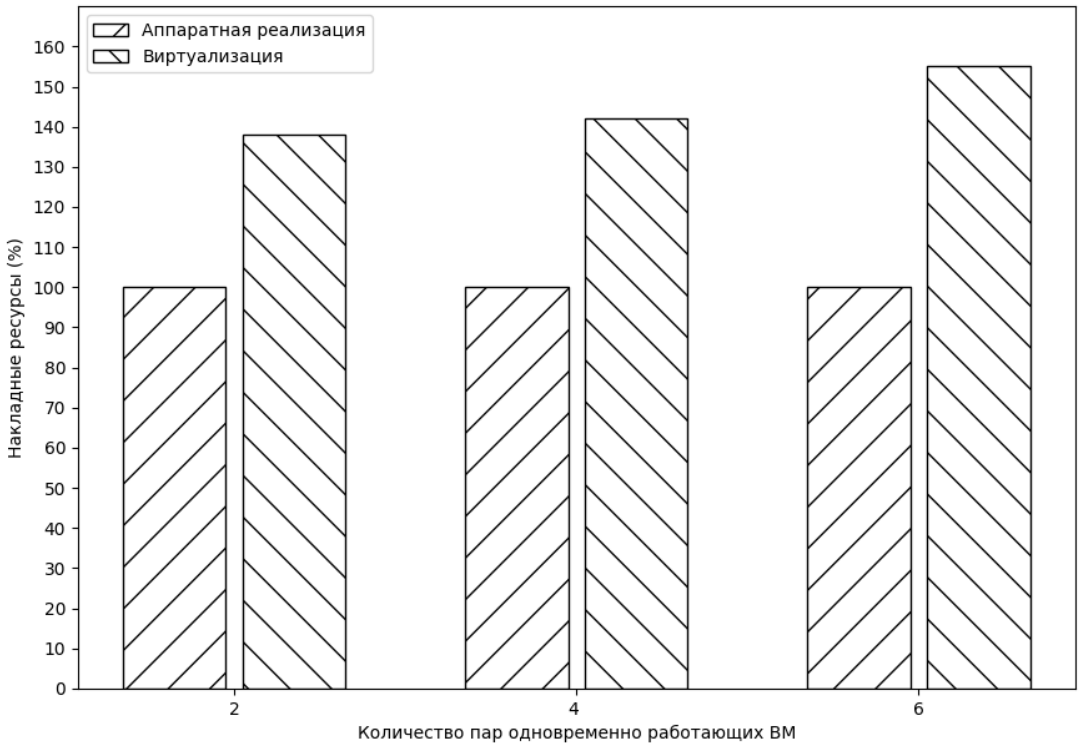

Рисунок 4 – Сравнение результатов выполнения пользовательских приложений (ARMv8), несколько пар ВМ

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

На рисунке 6 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 5 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:

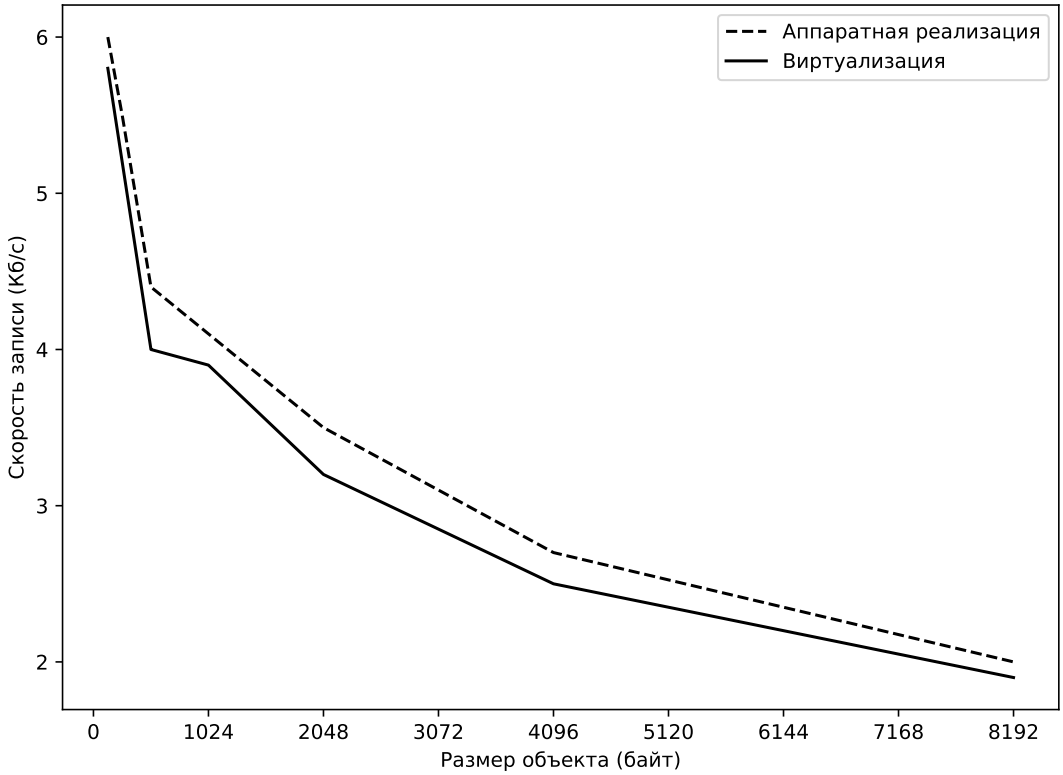

Рисунок 5 – Зависимость скорости записи на сервер MongoDB от размера объекта

— 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:

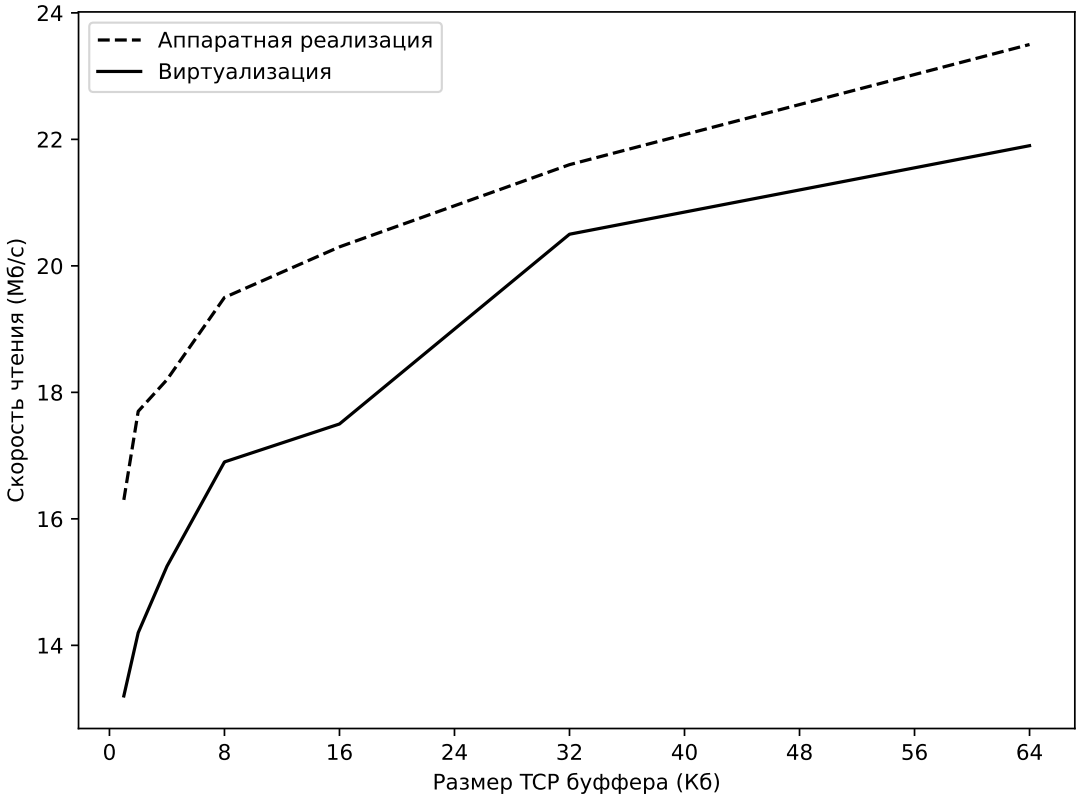

Рисунок 6 – Зависимость скорости чтения данных с сервера Apache от размера TCP буффера

- 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.

ЗАКЛЮЧЕНИЕ

В ходе выполнения научно исследовательский работы была достигнута ее цель – было проведено исследование эффективности и применимости метода программной реализации доверенной среды исполнения с помощью виртуализации процессоров архитектуры ARM.

Для достижения данной цели были решены следующие задачи:

- проведено исследование эффективности и применимости ПО;
- выполнено сравнение метода программной реализации с аппаратным методом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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)