Fachbereich Informatik

Vorlesung: Practice-oriented Security

Lehrende: Prof. Dr.-Ing. Sebastian Gajek

Benchmark: Confidential Cloud Computing for Datascience

Name: Lindner, Jannick
Matrikel-Nr.: 750286
Name: Otto, Christopher
Matrikel-Nr.: 610423

WS 2024/2025

Benchmark: Confidential Cloud Computing for Datascience

Christopher Otto[*]
Hochschule Flensburg
Flensburg, Germany

Jannick Lindner[†]
Hochschule Flensburg
Flensburg, Germany

ABSTRACT

Abstract. The increasing importance of data security has driven the adoption of confidential cloud computing technologies, such as Intel Trust Domain Extensions (TDX) and AMD Secure Encrypted Virtualization (SEV). This paper investigates the performance impact of enabling these technologies within the context of typical data science workloads on cloud infrastructures. Using benchmark tests on Azure virtual machines, including both confidential and non-confidential configurations (Das-v5, Dv5, DCas-v5, and DCes-v5), we evaluate the trade-offs between security and performance. The results indicate that enabling encryption introduces significant write latency without substantial read performance degradation, while certain workloads show varying degrees of performance loss depending on the hardware and security configurations. These findings provide a practical reference for IT decision-makers and data scientists considering confidential computing environments, balancing the need for security with computational efficiency.

Keywords: Encryption, Cloud Computing, Data privacy, Confidential Cloud Computing, Intel TDX, AMD SEV, Data Science, Benchmarking, Virtual Machines

Contents

[*] Author ID: 610423
[†] Author ID: 750286

LIST OF ABBREVIATIONS

ASP	AMD Secure Processor
KDC	Key Distribution Center
MAC	Message Authentication Code
SAML	Security Assertion Markup Language
SEV	Secure Encrypted Virtualization
SEV-ES	Secure Encrypted Virtualization - Encrypted State
SEV-SNP	Secure Encrypted Virtualization - Secure Nested Paging
SGX	Software Guard Extensions
SSO	Single Sign-On
TDs	Trust Domains
TDX	Trust Domain Extensions
TLS	Transport Layer Security
VMs	Virtual Machines
VMX	Virtual Machine Extensions

1 MOTIVATION

In today's digital world, companies and research institutions face the challenge of processing increasingly large amounts of data securely and efficiently. Data science applications, used in various fields such as finance, healthcare, and scientific research, require powerful and scalable infrastructures [9]. The cloud plays a central role in this, as it offers flexible and cost-efficient solutions to offload computationally intensive tasks [10, 3]. With the increasing use of sensitive data in the cloud, the need to ensure top-level data protection and security has grown. This is where technologies such as confidential computing come into play, using hardware-based approaches to ensure the confidentiality of data even during processing[21]. These technologies are particularly important in cloud environments, as they make it possible to ensure that no unauthorized third parties can access data, even when processed in public infrastructure[16]. In this context, Microsoft Azure has developed a range of virtual machines(VMs), known as confidential VMs, such as the DCas-v5 and DCes-v5 series, which offer encrypted data processing and additional security features[11].

Previous studies, primarily conducted by manufacturers such as Intel and AMD, have shown that the use of confidential VMs in practice often results in only minor performance losses. For example, Intel demonstrated with its fourth-generation Xeon processors and the implementation of Trust Domain Extensions (TDX) that the performance impact of security-focused partitioning is minimal[6][5]. Similarly, AMD's study on Google Cloud N2D Confidential VMs found that the use of SEV-SNP technology compared to non-confidential instances only results in slight performance degradation[1]. These studies provide an important foundation for understanding the performance of confidential VMs, but they also highlight a weakness: they are often based on synthetic data and workflows, without addressing real, practical use cases such as typical data science tasks. Therefore, there is a need for further research that examines the impact of confidential VMs on real applications and workloads in more depth[1][6].

The question that arises is whether the use of such confidential VMs results in significant performance losses in the specific context of data science applications, which rely on efficient data processing. This is of particular interest because the balance between security and performance is a critical factor for the practical use of such technologies. In this paper, we examine the performance differences between confidential VMs and non-confidential VMs in the Azure cloud. We focus on typical data science tasks and compare the smallest VMs from the Das-v5, DCas-v5, Dv5, and DCes-v5 series[11]. The aim of this investigation is to provide a solid foundation for deciding when and under what circumstances the use of confidential VMs is appropriate and to what extent potential performance losses are justifiable compared to the security benefits. The results of this study are intended to provide not only data scientists but also IT decision-makers with a well-informed basis for choosing VMs in security-critical application areas.

2 TDX EXPLANATION

Intel Trust Domain Extensions (TDX) is a technology developed by Intel that provides hardware-assisted virtualization and isolation to protect the processing of confidential data in virtual machines (VMs). The goal of TDX is to ensure the integrity and confidentiality of data in so-called Trust Domains (TDs), even if the hypervisor or other system layers may be compromised[4].

2.1 System Architecture

Intel TDX is based on a special hardware extension within the Intel Xeon processor family. This architecture consists of two main components: the TDX-enabled processor and the TDX module. The processor supports hardware-based virtualization, while the TDX module is a signed software module loaded into a special, isolated memory region (SEAM-RANGE). This module provides interfaces for hypervisor functions and allows the hypervisor to manage TDs securely without accessing their confidential memory or state[4][7].

2.2 Memory Isolation

TDX uses Virtual Machine Extensions (VMX) to enforce memory isolation for TDs. In TDX, hypervisors are no longer trusted for memory management, so the TDX module handles memory address translation and management of TDs' private memory areas. In addition to isolation, TDs' memory is cryptographically protected, preventing attacks on the physical hardware, such as by malicious DMA devices[4][7].

A physical memory page becomes part of the secure memory by enabling the so-called TD Owner Bit, which indicates that this page belongs to a TD and is protected. The memory controller ensures that only processes in SEAM mode can access these memory areas[4].

2.3 Memory Access and Protection

TDX not only protects access to memory but also ensures its integrity. Every write operation in a TD's memory is verified by a cryptographic MAC (Message Authentication Code), which ensures that no unauthorized modification has occurred[4].

2.4 TD Exit and Hypervisor Interactions

TDX allows TDs to use the hypervisor for certain I/O operations but with a critical limitation: TDs perform I/O operations through the TDX module process, which minimizes direct access to the TD's sensitive data. For example, during a TD exit, only the minimal necessary data is made available to the hypervisor[4].

2.5 Memory Management and Access to External Resources

To further enhance security, memory areas shared between TDs and the hypervisor are not cryptographically protected. The protection and confidentiality of data exchanged over these areas must be ensured through additional mechanisms, such as Transport Layer Security (TLS). This is particularly important for network and I/O communications[4].

2.6 Relevance of TDX for Confidential Computing

TDX provides the foundation for confidential computing in cloud environments by ensuring that both the integrity and confidentiality of data in virtual machines are preserved. This is especially relevant in environments where cloud service providers have full access to hardware and software but should not have insight into the protected data of tenants[4][7].

3 INFRASTRUCTURE FOR THE STUDY OF CONFIDENTIAL CLOUD COMPUTING

The experimental environment of our study is based on four different types of virtual machines (VMs) provided by the Microsoft Azure Cloud platform. The goal was to compare the performance of "confidential" and "non-confidential" cloud computing environments. We focused on the following VM types:

3.1 Non-confidential Cloud Computing

3.1.1 VM Type D2as_v5. The D2as_v5 instances belong to the series of general-purpose standard VMs on Azure, powered by AMD EPYC 7763v 3rd Generation processors. These instances offer high computing power but do not include specific confidentiality features such as hardware-based encryption. They are optimized for a variety of workloads requiring a flexible balance between computing power and memory but do not need special protection for sensitive data during processing[12].

Specifications of the D2as_v5 instance:

- Processor: AMD EPYC 7763v 3rd Generation.
- vCPUs: 2 virtual CPUs.
- RAM: 8 GB memory.
- Disk (OS): 50 GB Standard SSD.
- **Use case:** Standard workloads where confidentiality on a hardware level is not required, such as web servers, application development, or databases without sensitive information[12].

3.1.2 VM Type D2_v5. The D2_v5 instances utilize Intel Xeon Platinum 8370C processors from the Ice Lake series, which offer high performance but operate without specific hardware-based encryption or security features. These instances are suitable for a broad range of workloads that do not have specific security requirements but require maximum computational power[15].

Specifications of the D2_v5 instance:

- Processor: Intel Xeon Platinum 8370C.
- vCPUs: 2 virtual CPUs.
- RAM: 8 GB memory.
- Disk (OS): 50 GB Standard SSD.
- **Use case:** Ideal for workloads that require high computational performance without additional security requirements, such as batch processing, web servers, or application development[15].

3.2 Confidential Cloud Computing

3.2.1 VM Type DC2as_v5. The DC2as_v5 instances represent the "confidential computing" model and are also based on AMD EPYC 7763v 3rd Generation processors. These processors support Secure Encrypted Virtualization (SEV), which enables the encryption of data during processing at the hardware level. These additional security mechanisms ensure the confidentiality of data while it is being processed in the cloud[13].

Specifications of the DC2as_v5 instance:

- Processor: AMD EPYC 7763v 3rd Generation with SEV.
- vCPUs: 2 virtual CPUs.
- RAM: 8 GB memory.
- Disk (OS): 50 GB Standard SSD.
- **Use case:** Workloads that require secure data processing, such as financial applications, confidential data analysis, or applications where privacy regulations like GDPR play a role[13].

3.2.2 VM Type DC2es_v5. The DC2es_v5 instances are based on Intel Xeon processors and support Intel SGX (Software Guard Extensions), which provide hardware-based confidentiality protection. This security feature allows data to be processed within protected enclaves, preventing even the cloud provider from accessing the data during its processing[14].

Specifications of the DC2es_v5 instance:

- Processor: Intel Xeon with SGX support.
- vCPUs: 2 virtual CPUs.
- RAM: 8 GB memory.
- Disk (OS): 50 GB Standard SSD.
- **Use case:** Workloads that require the highest level of security, such as applications for processing highly sensitive data that require protection through Intel SGX[14].

4 INFRASTRUCTURE COMPARISON

The key differences between these VM types lie in the processor architectures used and the security features specific to the confidential instances (DC2as_v5 and DC2es_v5). While the D2as_v5 and D2_v5 instances do not offer special protection mechanisms, the DC2as_v5 and DC2es_v5 instances support hardware-based encryption and confidential data processing. These functions ensure that sensitive data remains protected during processing, which is critical for certain workloads[11].

4.1 Security Features Comparison

- **D2as_v5 & D2_v5:** No special protection for data during processing. Data could theoretically be accessed by privileged attackers within the cloud infrastructure[12][15].
- **DC2as_v5:** Hardware-based encryption via AMD SEV ensures that even with memory access, no plaintext data can be viewed[13].
- **DC2es_v5:** Data protection through Intel SGX, enabling isolated and encrypted processing within secure enclaves, even against access by the cloud provider[14].

This difference in security mechanisms could impact performance, as the additional security layers in the DC2as_v5 and DC2es_v5 instances may introduce some overhead, affecting processing speed[11].

5 BENCHMARK METHODOLOGY

This section highlights the basics of benchmarks, what can go wrong, and what you have to keep in mind when designing or executing a benchmark.

5.1 Reproducibility

The main focus of a benchmark should always be to deliver consistent, reproducible and reliable results. The Reproducibility is without a doubt the most important factor when it comes to benchmarks. If a benchmark cannot be reproduced, the results could always come into question after a result is published. And without the confirmation of third parties, things like inconsistencies or errors are difficult to detect. If a benchmark can be easily recreated and always leads to the same results, it is executable by everyone with the available resources and

5.2 Environment Variables

The environment matters a lot when it comes to benchmarks. What services or programs are run in the background? What type of storage device does a PC use? SSD? HDD? eMMC? Is the storage soldered or is it serviceable by the user? What manufacturer made the memory chips? What revision is the hardware? Just by having another BIOS/UEFI version or settings, the CPU could draw more power, turbo higher or for a longer duration or just behave completely differently.

5.3 Run-to-run Variance

Run-to-run variance refers to the differences in results when the same benchmark is executed multiple times under supposedly identical conditions. In the context of benchmarking, especially in performance testing of software, hardware, or algorithms, this variance can reflect inconsistencies due to factors like system noise, resource contention, or randomization within the system or environment. A high variance makes it difficult to trust the results of a single run, as the performance metrics could change drastically each time - making the benchmark run unreliable. Also, if the variance is large, comparing two systems or configurations becomes challenging because the performance range overlaps. Small differences could be statistically irrelevant. The most important things to eliminate run-to-run variance are:

- **Isolate benchmarking environment:** Ensure no background processes are running that can interfere with the benchmark.
- **Use consistent workloads:** If the workload has random elements, ensure it is controlled to be as deterministic as possible.
- **Run benchmarks multiple times:** By running a test multiple times, we can average the results and analyze the variance statistically to understand if differences are meaningful.
- **Pin CPU and memory:** Restrict the benchmark to specific cores or isolate memory usage to reduce interference from other processes.

5.4 The biggest challenge (you)

Like in the previous section explained, the environment is very important when it comes to benchmarks. This is why it is very important to control it as best as possible. And when it comes to controlling things and consistency, one factor is always the one which is the hardest to control and the most likely to result in inconsistencies or errors - the human factor. The more manual input is required to execute a benchmark, the more room there is for human error. Sometimes even just entering a command again, clicking on a specific thing, installing another library or even running the same commands but in a different order can lead to a huger discrepancy in the results. The timing between tests also matters, because the CPU turbo durations might, or might not, reset by the time the next test starts, leading to a very large run to run variance. This is why our benchmark was designed to automate as many steps as possible to make the benchmark as safe, controlled and consistent as is feasible.

5.5 Can't control everything

While it is a good target to aim at, being able to control every variable is just not something you can do. Even when comparing two identical machines, there are some performance differences that will be measurable even if the machines appear to be 1:1. In a two separate Videos, Linus Tech Tips and Gamersnexus discovered, that performance of a processor can vary just because of manufacturer tolerances and the silicon lottery that 2 people with the same experiments could come to very different conclusions even when executing exactly the same workloads/benchmarks:

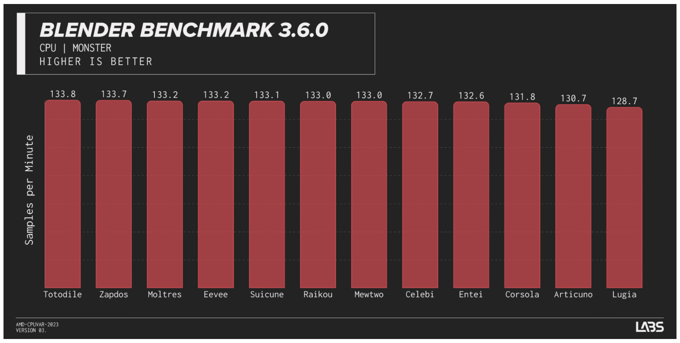

Figure 1: Linus Tech Tips 7800X3D Comparison [20]

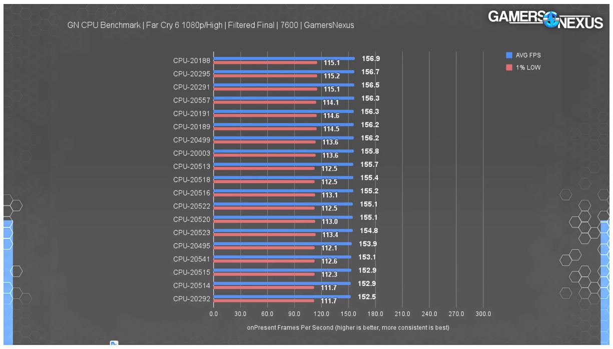

Figure 2: Gamers Nexus Video [17]

5.6 Theory

Our Theory is that, because TDX and SEV rely on Instructions from the CPU to de- and encrypt the memory, that the memory latency or the latency of specific tasks will be impacted. Because of this, we choose some tasks which will test this hypothesis and either prove it to be false or true.

6 BENCHMARKS

In this section we will highlight some of the benchmarks we choose and explain what they do and why we chose them.

6.1 Standard Run

Our standard run, which is written down in the benchmark-plan.txt consists of the following benchmarks:

```
# Format: path_to_benchmark,type,datasets,waittime_in_seconds,num_executions
systeminfo,standard,,5,1
fio_4k_sequentialread,standard,,10,1
fio_4k_sequentialwrite,standard,,10,1
fio_4k_randomread,standard,,10,1
fio_4k_randomwrite,standard,,10,1
fio_64k_sequentialread,standard,,10,1
fio_64k_sequentialwrite,standard,,10,1
fio_64k_randomread,standard,,10,1
fio_64k_randomwrite,standard,,10,1
fio_1m_sequentialread,standard,,10,1
fio_1m_sequentialwrite,standard,,10,1
fio_1m_randomread,standard,,10,1
fio_1m_randomwrite,standard,,10,1
SimplePrime.ipynb,kaggle_notebook,,20,5
credit-fraud-dealing-with-imbalanced-datasets.ipynb,kaggle_notebook,...,20,1
twitter-sentiment-analysis.ipynb,kaggle_notebook,...,20,1
pneumonia-detection-using-cnn-92-6-accuracy.ipynb,kaggle_notebook,...,20,1
lstm-sentiment-analysis-keras.ipynb,kaggle_notebook,...,20,1
getting-started-with-a-movie-recommendation-system.ipynb,kaggle_notebook,.,20,2
airline-delay-notebook.ipynb,...,20,1
e-commerce-eda-purchase-classification.ipynb,kaggle_notebook,...,20,1
google-play-store-analysis.ipynb,kaggle_notebook,...s,20,1
creditcard-fraud-balance-is-key-feat-pycaret.ipynb,kaggle_notebook,...,20,1
energy-eda-segmentation-and-prediction.ipynb,kaggle_notebook,...,20,3
pyperformance default,standard,,10,1
```

As displayed in the code above, the benchmark starts with some different size Fio benchmarks (after collecting system information) in sequential and random read or write operations. Each benchmark targets a duration of 1 minute with a 20-second break in between each benchmark. After the Fio benchmarks, the Kaggle Notebooks get executed after one another. Some notebooks are executed more than once to increase consistency and confirm the run-to-run variance. Lastly, the PyPerformance "default" benchmark suite gets executed, this is a suite which incorporates a number of benchmarks. For more details on the benchmarks, and the benchmark-plan, you can check out the code in our repository.

6.2 PyPerformance

The PyPerformance Benchmark focuses on real world benchmarks. To do this, the Benchmarks all use a virtual environment to control the environment variables as best as possible [19]. PyPerformance features around 44 benchmarks, which all serve a different purpose. They all cover a specific usecase or execution type. From crypto_pyaes, which is a benchmark, of a pure-Python implementation of the AES block-cipher in CTR mode using the pyaes module to large floating point operations, the regex_v8 benchmark, which features all regexp operations from 50 of the most popular pages on the web or the raytrace benchmark, which generates an image using a simple raytracing rendering technique [18]. We choose PyPerformance because of the focus on real-world approach which was used to create these benchmarks, as well as the number and the consistency.

6.2.1 Metrics. For Pyperformance we measure only one metric - **execution time**. Which means the faster, the better. So a smaller number or smaller bar in a chart is a better result. All execution times are a mean across many executions.

6.3 Fio

Fio stands for Flexible IO Tester. This Library can be used to test all types of disk drives (SSDs or HDDs) without having to manually write instructions. We used Fio for the ease of use, flexibility and metrics Fio offers. [2] Moreover, Fio was also used to benchmark the performance impact of LUKS previously. [8]

6.3.1 Metrics. For FIO there are 3 Metrics we can look at that are relevant:

- - **Throughput** (Data transfer rate)
- - **IOPS** (Input/output operations per second)
- - **Latency** (how long does the drive take to respond)

IOPS are strongly related to the throughput when only one type of operation is executed (read only, write only). In a Fio output file, there are multiple measures of latency; clat, slat and lat:

- **clat:** (completion latency in nanoseconds): This metric represents the time taken from the submission of a request until it is completed by the storage device. It's measured in nanoseconds and gives insight into how long the storage takes to process requests.
- **lat:** (latency in nanoseconds): This usually refers to the overall latency for a particular operation, which can include additional overhead. The lat may take into account both the queuing time (how long the request sits waiting before it is processed) and the completion time.
- **slat:** (submission latency in nanoseconds): This refers to the time it takes for the operating system or application to submit an I/O request to the storage device.It is measured from the point when the I/O operation is requested until it is actually handed over to the storage device for processing.A high slat indicates that the request is spending more time in the submission queue or waiting for the OS to pass it on to the device.

For all benchmarks when speaking of fio latency, we always speak about the **completion latency (clat)**.

6.4 Kaggle Notebooks

Kaggle is a site which features many data-science and machine learning related python notebooks. We used Kaggle because we wanted real world code examples of what a workload of an ML Engineer or Data Scientist might look like. Some of the notebooks have been modified to use updated commands because of their age, are reduced in processing intense tasks like ML epochs or reduced batch sizes and could be missing features. This is because when running them unmodified, the complete benchmark suite would take very long (more than a day) to complete. Also, the imports have all been updated to account for the changed project structure with all datasets in the datasets folder apart from the notebooks themselves. The Kaggle Notebooks are the "heart" of our benchmark, they are what truly sets this benchmark apart from others, besides the ease of use and the configurability. Since the notebooks from kaggle feature operations and code snippets an actual data scientist or machine learning engineer would use in real life scenarios, this data is invaluable compared to synthetic benchmarks, which might be more consistent but fail to recreate an authentic experience, compromising size constraints and ease of use or speed of execution for accuracy.

6.4.1 How it works. The Kaggle Notebooks all get executed by the manage.py python script. This script then uses nbconvert to convert the notebooks and executes them, meaning all the code cells inside the notebook are executed after each other. A Jupyter or Kaggle notebooks consists of code, which is structured in cells. Although you could put all the code in one single cell, notebooks are often divided into cells with specific purposes such as reading data, importing libraries, restructuring data, training models, verifying models or just creating and displaying graphs. These cells are usually executed after each other, while the user can wait to execute the next cell or make changes. As long as the notebook is open, and the kernel is active, all executed code, meaning variables, imported libraries and functions are kept in memory and the cells have access to functions in other cells given that they were previously executed in this session. While it would be theoretically possible to execute the notebook from the last to the first cell, this is often not possible since the imports, data and functions used in the last cells are often created in previous cells.

6.4.2 Metrics. Similar to Pyperformance, for the Kaggle Notebooks we measure only one metric - **execution time**. Because the Notebooks are very different and all perform other tasks which might stress other components, there is no way to measure other metrics across all the notebooks. This measurement is taken by the manage.py script.

```python
import time
def find_primes(n):
    start_time = time.time()
    primes = []
    num = 2


    while len(primes) < n:
        is_prime = True

        # Check if the number has divisors other than 1 and itself
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                is_prime = False
                break

        if is_prime:
            primes.append(num)

        num += 1

    # Print the prime numbers to the console
    for prime in primes:
        print(prime)

    end_time = time.time()
    time_taken = end_time - start_time
    print(f"\nTime taken to find {n} prime numbers: {time_taken:.4f} seconds")
```

```

find_primes(100000)
```

6.4.3 SimplePrime.ipynb.

This notebook was written by us as a quick test and prints the first 100,000 prime numbers. This is also a validation test for our method of executing the notebooks, since this workload is very consistent and should always take the same amount of time on the same machine, since the number of calculations and comparisons are the same every time, because the first 100,000 prime numbers will always stay the same. We will later discuss why this is important in the conclusion.

6.4.4 airline-delay-notebook.ipynb. Data analysis and predictions on flight delays using historical data from 2015.

6.4.5 credit-fraud-dealing-with-imbalanced-datasets.ipynb. The Kaggle notebook "Credit Fraud Detector" focuses on building and evaluating predictive models to detect fraudulent transactions in a highly imbalanced dataset, where fraud makes up only a small fraction of the total transactions.

6.4.6 creditcard-fraud-balance-is-key-feat-pycaret.ipynb. Similar to the previous notebook in this notebook, the focus is on addressing data imbalance in binary classification problems, particularly through the use of oversampling and undersampling techniques. These methods are introduced to compensate for situations where one class (e.g., fraud vs. non-fraud) is significantly under-represented in the data.

6.4.7 e-commerce-eda-purchase-classification.ipynb. This Notebook aims to classify customers and purchases. To do this it looks at a dataset from an online shopping retailer and investigates the correlation between no. of page visits or length of a visit with a purchase decision.

6.4.8 energy-eda-segmentation-and-prediction.ipynb. The Kaggle notebook focuses on a project analyzing energy consumption data from the Netherlands, specifically using the 2013 dataset from the Coteq company. The analysis involves two main tasks: estimation and segmentation of energy consumption based on zip codes over the years.

6.4.9 getting-started-with-a-movie-recommendation-system.ipynb. Like the name suggests, this notebook is using user data from streaming services to suggest new movies using the watch history or previously seen movies of the user as input to make predictions. It basics of Recommendation Systems and demonstrates how to build a simple movie recommender using the TMDB 5000 Movie Dataset by using Demographic, Content Based and Collaborative Filtering.

6.4.10 google-play-store-analysis.ipynb. This Kaggle notebook analyzes a dataset of over 2.3 million mobile apps from the Google Play Store. The notebook explores various aspects of the mobile apps on Google Play, including ratings, installs, and trends focusing on app metrics and trends, with a particular emphasis on aiding analysis for developers and researchers.

6.4.11 lstm-sentiment-analysis-keras.ipynb. In this Kaggle notebook, the author Peter Nagy focuses on sentiment analysis using a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) layers to classify tweets as positive or negative. This notebook is an introductory-level example of sentiment analysis using LSTM, offering basic functionality but limited by its imbalance in data and lack of further enhancements.

6.4.12 pneumonia-detection-using-cnn-92-6-accuracy.ipynb. This notebook focuses on pneumonia detection using chest X-ray images through the application of a Convolutional Neural Network (CNN).

6.4.13 twitter-sentiment-analysis.ipynb. Similar to the LSTM Sentiment analysis notebook, this notebook uses Twitter data to train a model to analyze and predict the sentiment of tweets such as: "I love the music", "I hate the rain" or "i don't know what i'm doing". It also uses an LSTM approach. This Kaggle notebook implements a natural language processing (NLP) model to perform sentiment analysis on text data. It also uses a lot of visualizations to evaluate the model.

7 ANALYSIS

In this section, we will display graphs and show our findings while highlighting important differences or outliers in the data. While we will emphasize some aspect of the data, we won't draw conclusions from it until the next chapter - the conclusion. To make the results more readable and understandable, the machines were relabeled as follows:

Machine	New Name	Features
DC2eds_v5	TDX-Encrypted	TDX, Disk Encryption
DC2eds_v5_noDisk	TDX-Non-Encrypted	TDX
D2_v5	Intel-Non-TDX	None
DCas_v5	SEV-Encrypted	SEV, Disk Encryption
DC2as_v5_noDisk	SEV-Non-Encrypted	SEV
D2as_v5	AMD-Non-SEV	None

Table 1: Test Machines

7.1 PyPerformance

First, we're going to be looking at some general performance numbers, too do this we will use some selected PyPerformance Benchmarks.

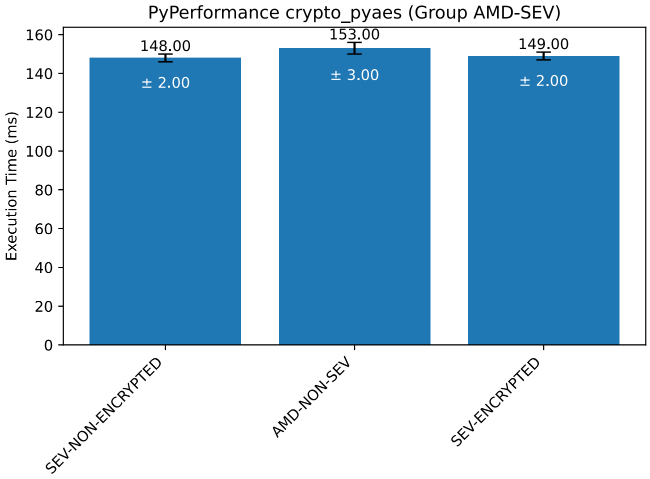

Figure 3: PyPerf Performance: Crypto PyAES on AMD-SEV

As expected from a Pyperformance benchmark, the crypto results are very consistent and even if there are some minor differences in the numbers the general execution time around 150ms of the machines are all within standard deviation of each other, so there isn't really a significant difference in either direction.

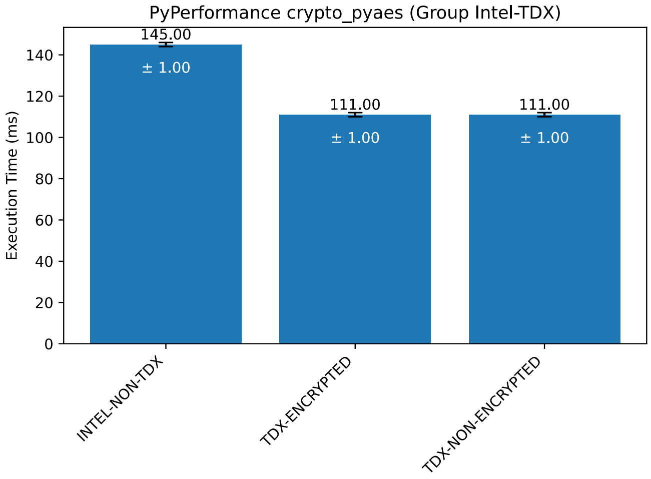

Figure 4: PyPerf Performance: Crypto PyAES on Intel-TDX

Unlike the AMD-based benchmarks, the Intel-based ones show an outlier - with the non-TDX Machine performing significantly slower or worse than both the TDX ones. Although this is expected behavior, since the non-TDX machine is not the 1:1 CPU equivalent to the TDX machines as explained in the infrastructure section.

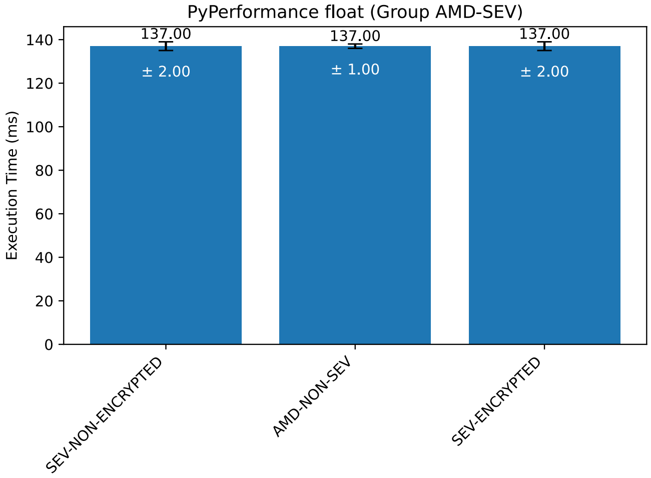

Figure 5: PyPerf Performance: Float on AMD-SEV

Continuing with the trend of very consistent results and results that are close to each other, this benchmarks shows how consistent the results of the benchmark truly are, with only the standard deviation making a difference between the machines.

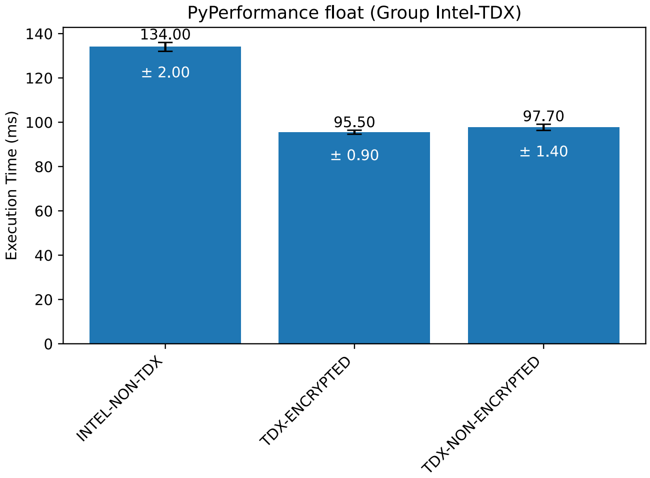

Figure 6: PyPerf Performance: Float on Intel-TDX

While the non-encrypted Machine is slightly faster in this benchmark, the difference between the encrypted and the non-encrypted machine is too small to make it significant. The non-TDX machine continues to perform about 25-30% worse than the TDX machines.

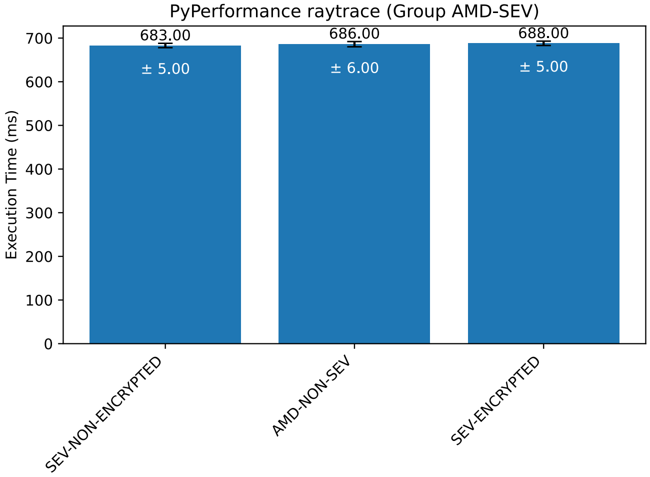

Figure 7: PyPerf Performance: Raytrace on AMD-SEV

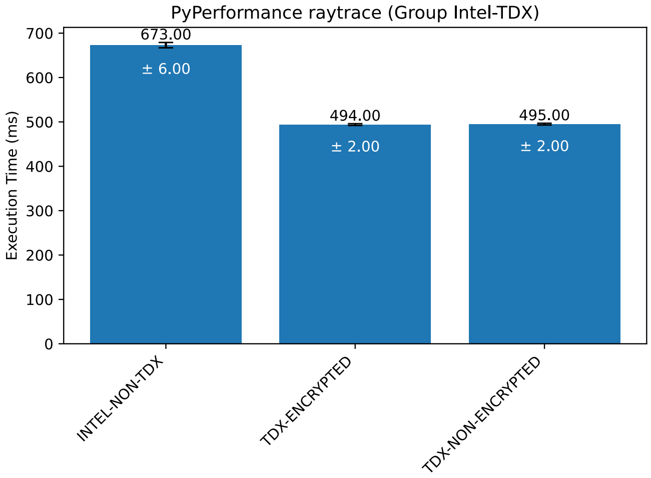

Figure 8: PyPerf Performance: Raytrace on Intel-TDX

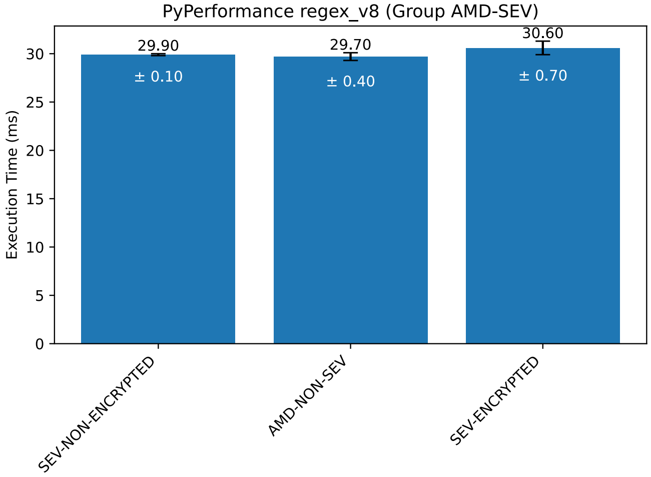

Figure 9: PyPerf Performance: Regex V8 on AMD-SEV

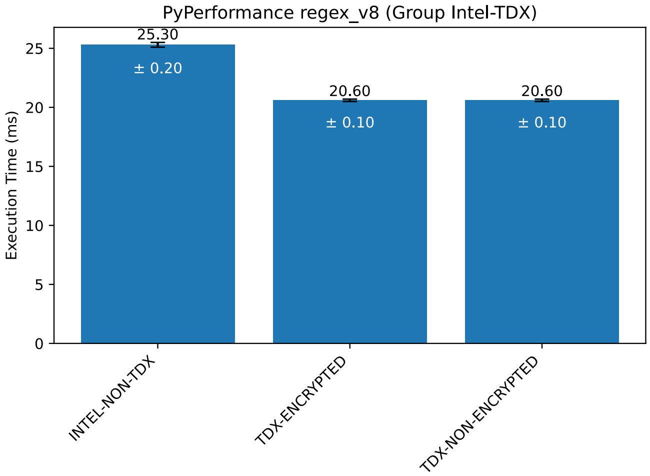

Figure 10: PyPerf Performance: Regex V8 on Intel-TDX

7.2 Fio

Next, before taking a look at the "heart" of our benchmark, the Kaggle notebooks, we will take a look at the Fio benchmarks.

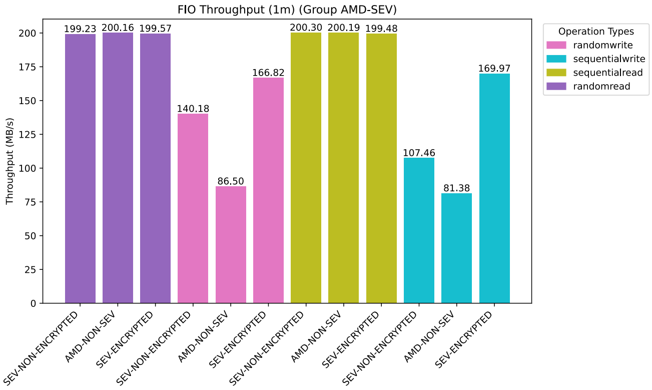

Figure 11: FIO Throughput for 1M Group on AMD-SEV

While the read benchmarks do not seem to discover any differences between the machines and only show a bottleneck at 200 MB/s, which is probably the Interface transfer limit, the write-benchmarks show a difference, with this difference getting larger when the files are random instead of sequential.

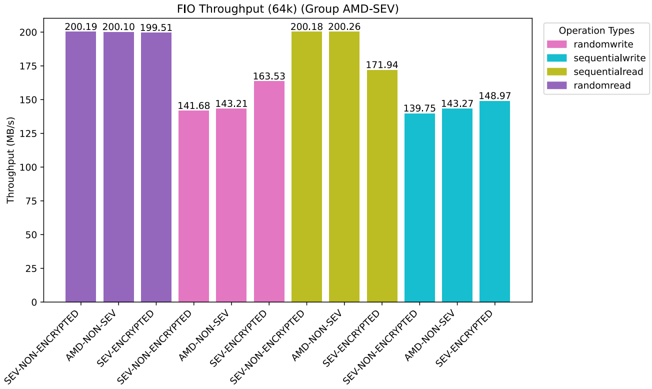

Figure 12: FIO Throughput for 64k Group on AMD-SEV

Unlike with the larger filesize, the smaller 64k files do not seem to have as much of an impact on the performance depending on the machine. The only odd result is the slower sequentialread result of the SEV-encrypted machine.

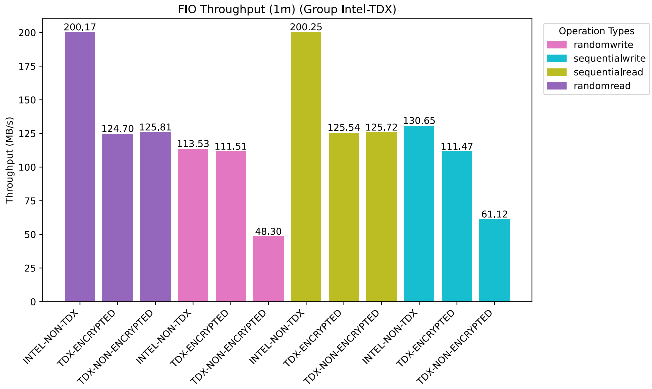

Figure 13: FIO Throughput for 1M Group on Intel-TDX

The intel benchmarks show a large performance difference between the TDX and non-TDX machines. Especially, the NON-TDX machine seems to be faster with reads, while the TDX non-encrypted machine seems to be significantly slower in writes than the encrypted TDX machine.

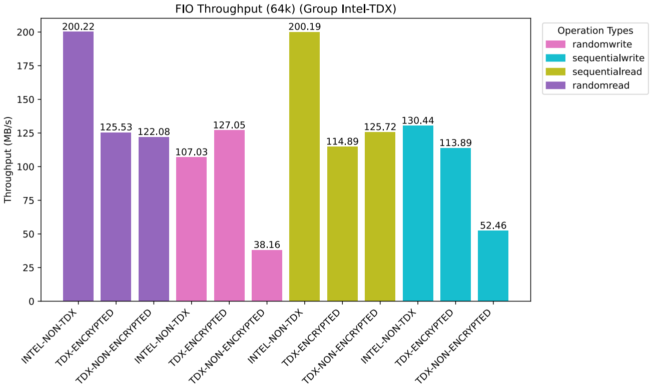

Figure 14: FIO Throughput for 64k Group on Intel-TDX

With the smaller file sizes, the benchmark just pretty much confirms our findings from the larger files.

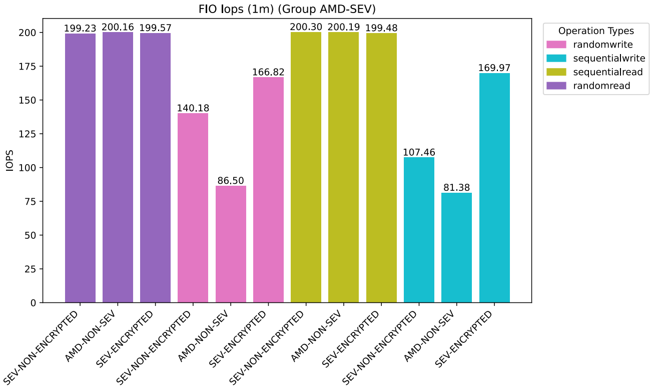

Figure 15: FIO IOPS for 1M Group on AMD-SEV

When comparing the Fio IOPS chart to the Figure 11 the connection between throughput and IOPS becomes very clear, especially when looking at it with a 1M file size. Since the IOPS and throughput charts are basically the same, we can skip the rest of the IOPS charts.

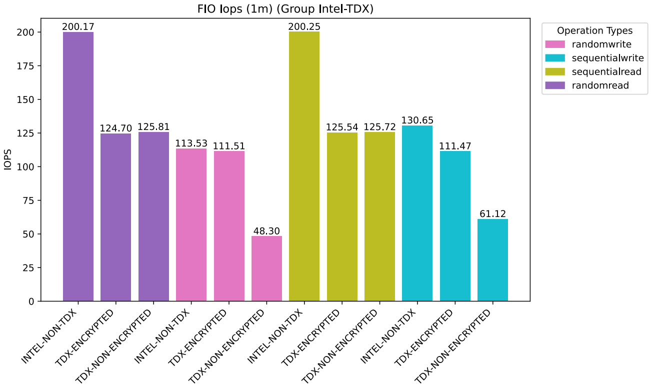

Figure 16: FIO IOPS for 1M Group on Intel-TDX

Like with the AMD-based charts, IOPS and throughput are the same when looking at Figure 13.

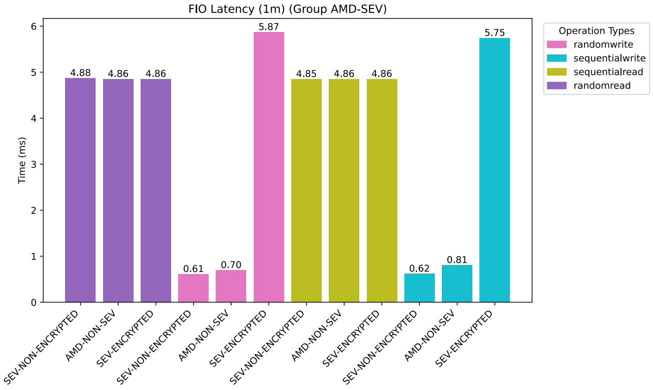

Figure 17: FIO Latency for 1M Group on AMD-SEV

While the read latency is basically the same across all 3 machines, there is an interesting pattern when looking at the SEV-Encrypted machine. The encrypted machine has a roughly 7-times higher latency when writing data - no matter if random or sequential.

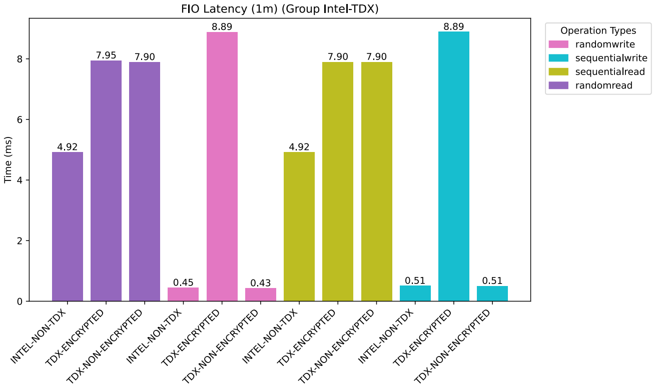

Figure 18: FIO Latency for 1M Group on Intel-TDX

Similar to the SEV encrypted machine, the TDX encrypted machine has significantly higher write latency (roughly factor 15) than the non-encrypted machines. So while the factor is higher, the trend is the same between Intel and AMD machines.

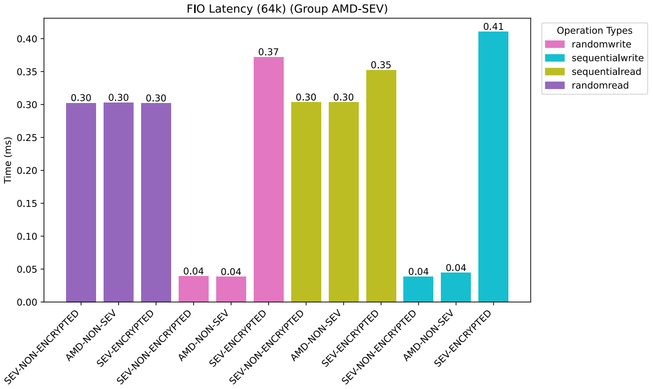

Figure 19: FIO Latency for 64K Group on AMD-SEV

For smaller filesizes, the trend continues, but the factor has risen from 7̃ to 1̃0.

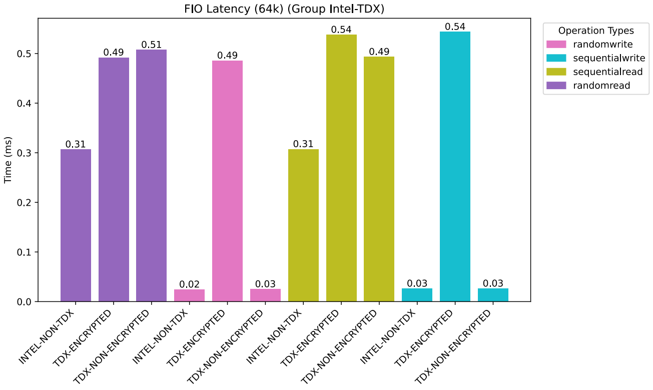

Figure 20: FIO Latency for 64K Group on Intel-TDX

For smaller filesizes on Intel, the trend continues, but the factor has risen slightly from 1̃5 to 1̃8.

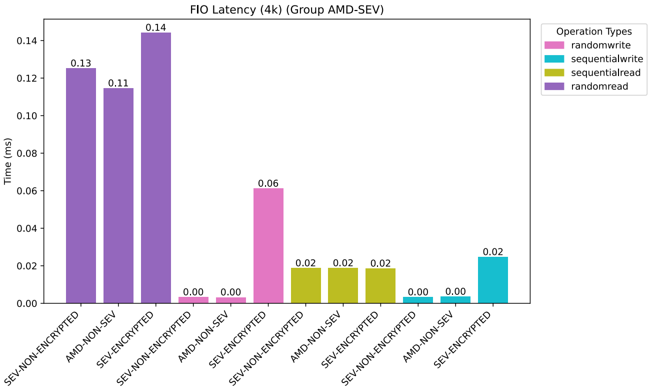

Figure 21: FIO Latency for 4K Group on AMD-SEV

While we can assume that the trend continues with smaller and smaller files, at 4k the latency is that small that we reach the limits of our resolution. The latency for encrypted storage is still higher, but the overall latency is very small. But the smaller filesize means we can measure the difference between sequential and random reads.

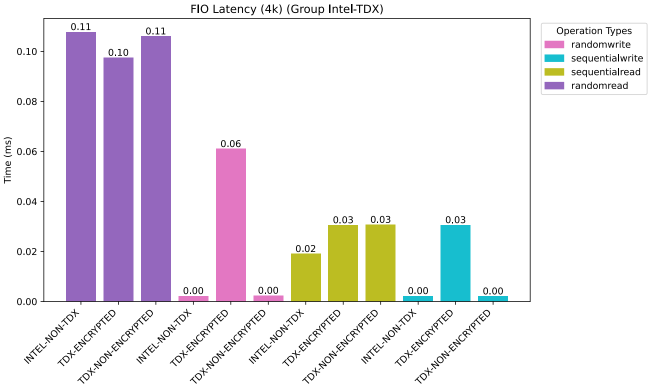

Figure 22: FIO Latency for 4K Group on Intel-TDX

The Intel group shows the same at 4k resolution as the AMD Group.

7.3 Kaggle Notebooks

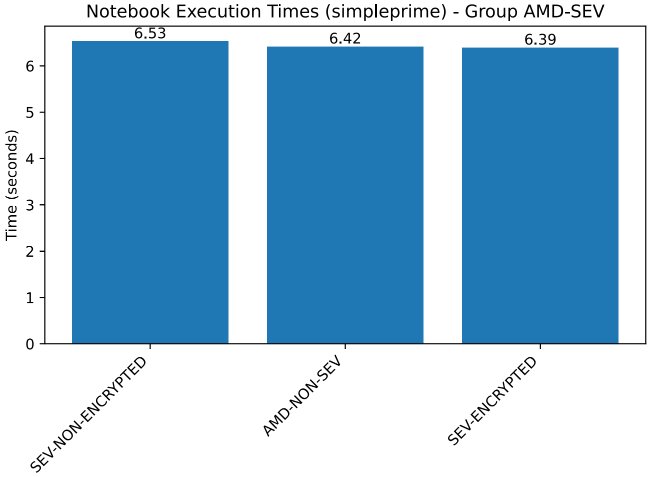

Figure 23: Simple Prime Benchmark on AMD Systems

The Simpleprime benchmark on AMD shows all results near identical to each other with very little deviation from each other.

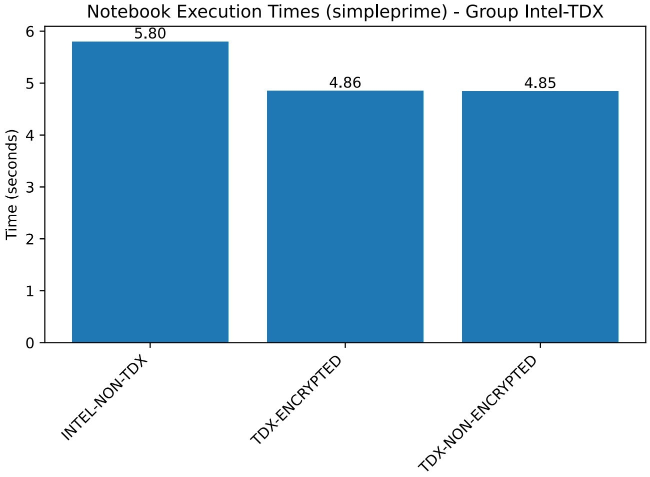

Figure 24: Simple Prime Benchmark on Intel Systems

Fitting in with the PyPerformance benchmarks, the Simpleprime benchmark shows that the non-TDX machine is slower than the TDX-Machines. The non-TDX machine has about 85% the performance in this benchmark of a TDX-Machine.

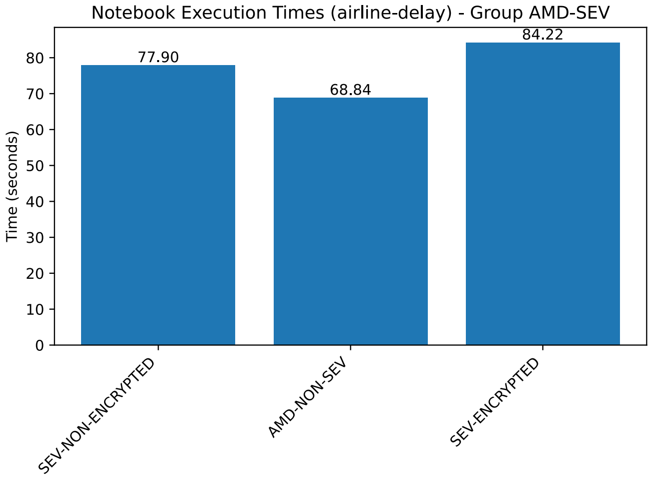

Figure 25: Airline Delay Benchmark on AMD Systems

The airline-delay benchmark shows the non-SEV machine as the fastest one, followed by the non-encrypted, and finally the encrypted one.

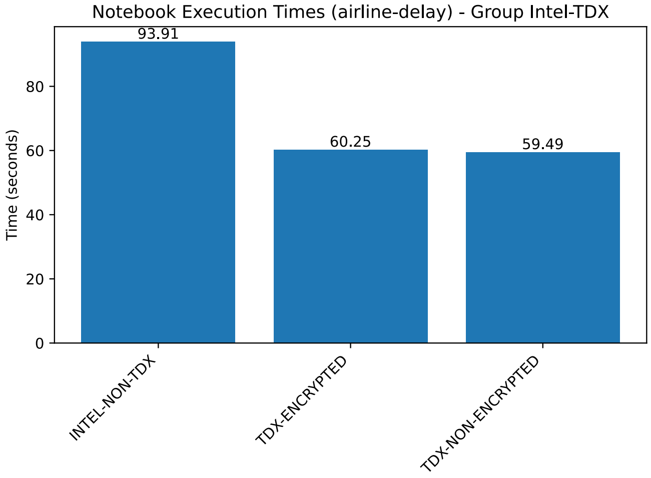

Figure 26: Airline Delay Benchmark on Intel Systems

For Intel, while the TDX-based machines performed near idential the non-TDX one only had around 65% of the performance.

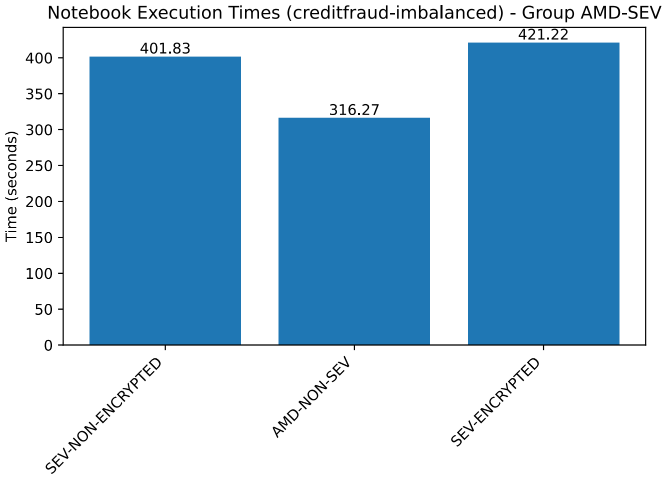

Figure 27: Credit Fraud (Imbalanced) Benchmark on AMD Systems

The imbalanced Credit Fraud benchmark shows the non-SEV machine as the fastest one, followed by the non-encrypted, and finally the encrypted one, continuing the trend of the airline-delay notebook.

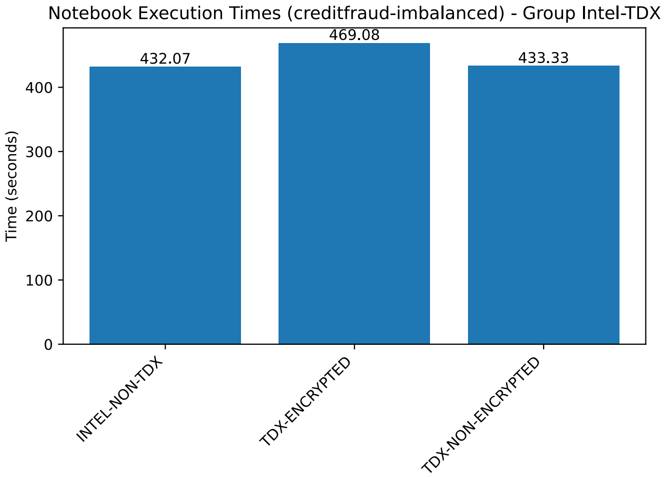

Figure 28: Credit Fraud (Imbalanced) Benchmark on Intel Systems

On Intel all machines performed nearly the same, with the TDX-encrypted machine being slower than the two others.

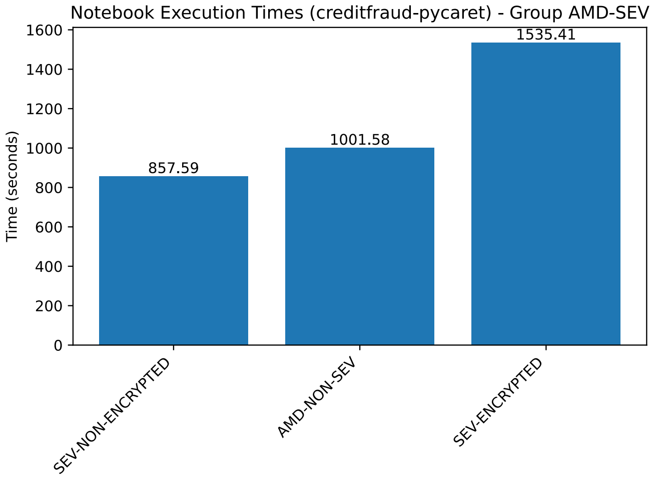

Figure 29: Credit Fraud (PyCaret) Benchmark on AMD Systems

The pycaret creditfraud benchmark shows a huge deviation from machine to machine.

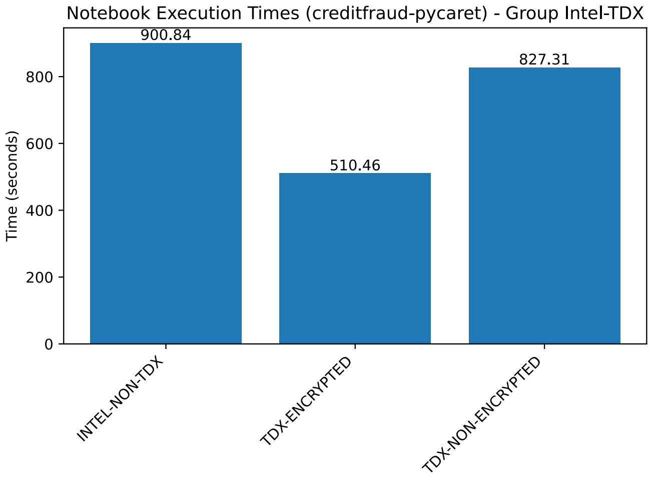

Figure 30: Credit Fraud (PyCaret) Benchmark on Intel Systems

Just like on the AMD-based Machines, the pycaret creditfraud benchmark shows a huge deviation from machine to machine.

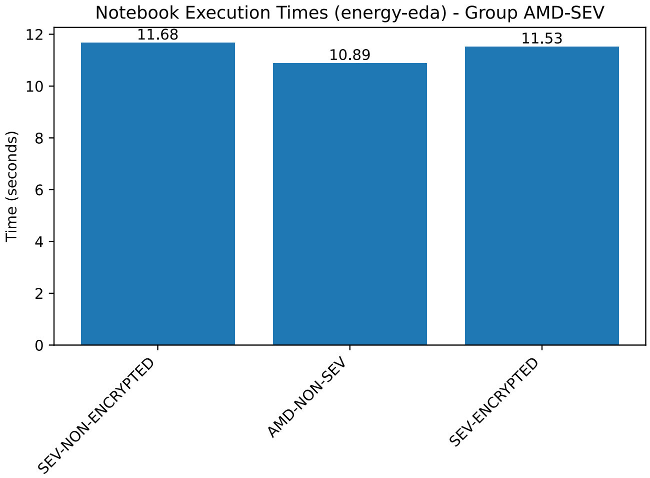

Figure 31: Energy EDA Benchmark on AMD Systems

In this benchmark all machines performed nearly the same, with the non-SEV machine being slightly faster than the two others.

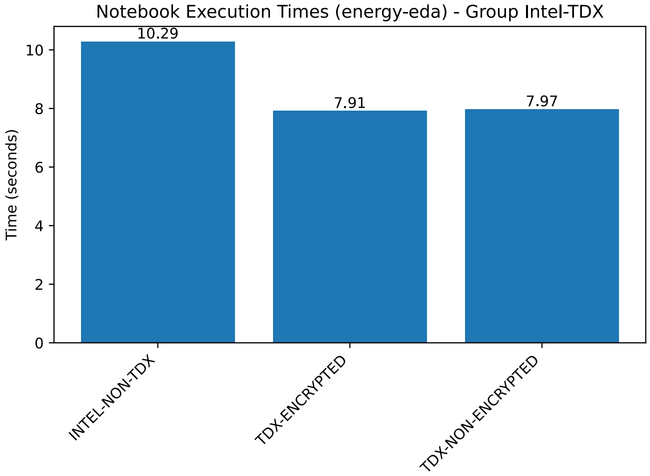

Figure 32: Energy EDA Benchmark on Intel Systems

Similar to the Pyperformance Benchmarks this notebook shows the non-TDX Machine having about 75% of the TDX ones while the TDX ones perform nearly identical to each other.

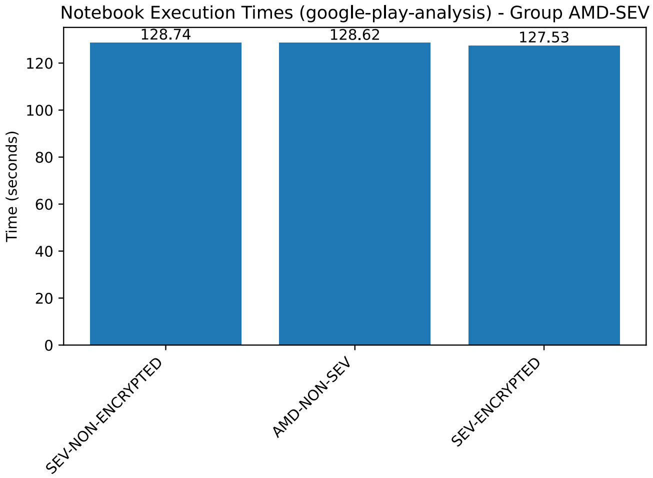

Figure 33: Google Play Analysis Benchmark on AMD Systems

In this benchmark all AMD machines performed the same with minimal variation.

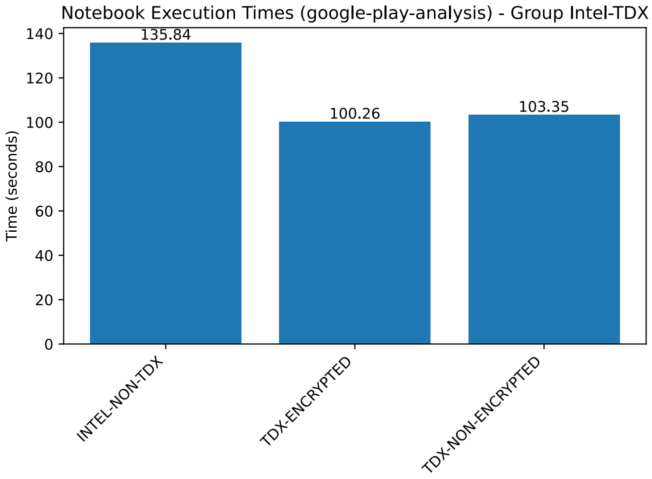

Figure 34: Google Play Analysis Benchmark on Intel Systems

Again, this notebook shows the non-TDX Machine having about 75% of the TDX ones while the TDX ones perform nearly identical to each other.

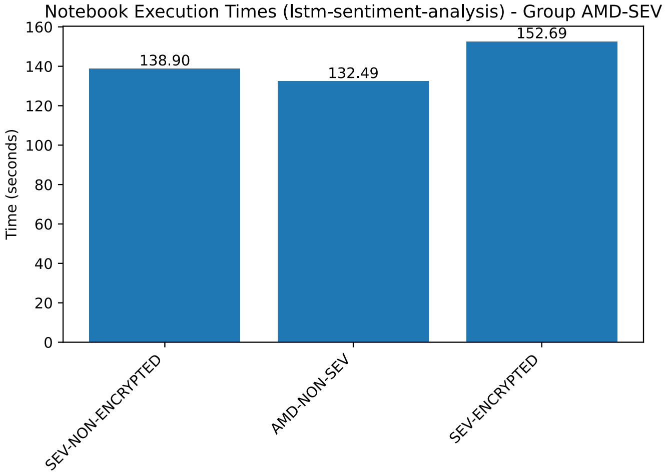

Figure 35: LSTM Sentiment Analysis Benchmark on AMD Systems

In this benchmark the non-SEV machine is the fastest one, followed by the non-encrypted, and finally the encrypted one, continuing the trend of the airline-delay and imbalanced credit fraud notebook.

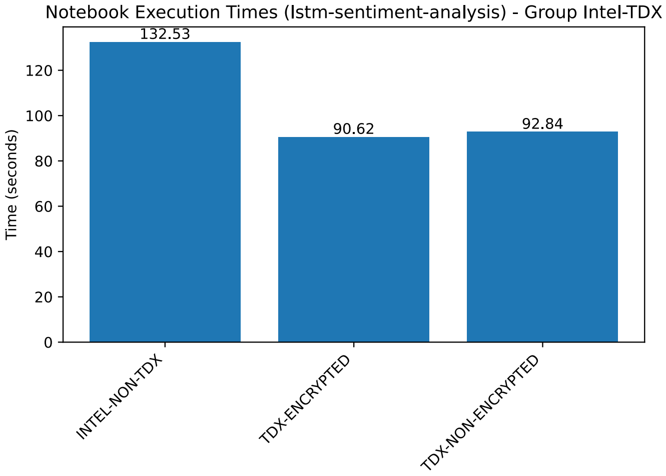

Figure 36: LSTM Sentiment Analysis Benchmark on Intel Systems

This notebook shows the non-TDX Machine having about 75% of the TDX ones while the TDX ones perform nearly identical to each other.

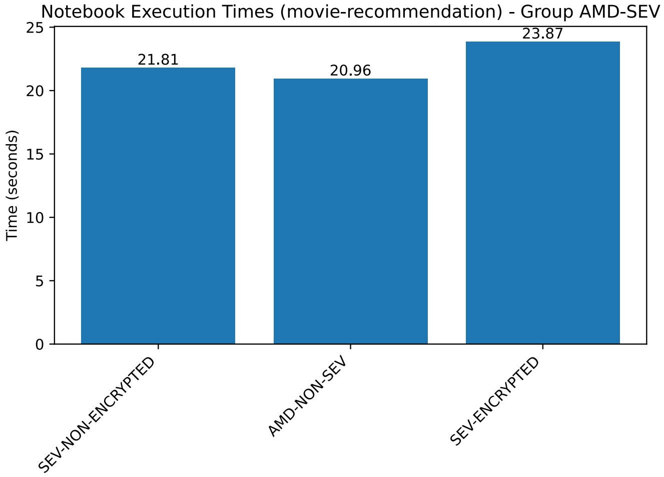

Figure 37: Movie Recommendation Benchmark on AMD Systems

In this benchmark the non-SEV machine is the fastest one, followed by the non-encrypted, and finally the encrypted one, continuing the trend of the airline-delay, imbalanced credit fraud and LSTM Sentiment Notebook.

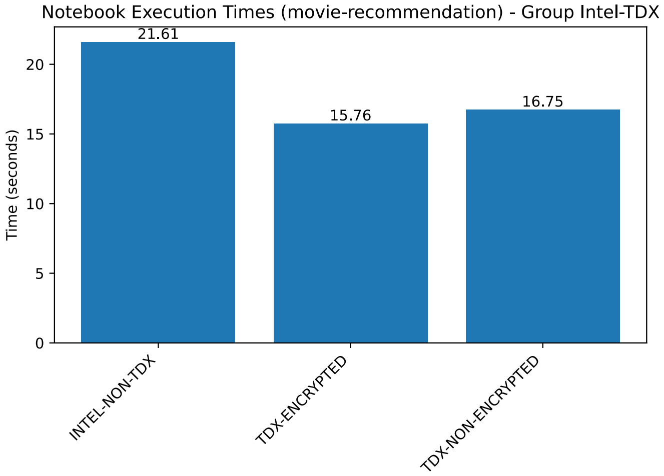

Figure 38: Movie Recommendation Benchmark on Intel Systems

This notebook shows the non-TDX Machine having about 75% of the TDX ones while the TDX ones perform nearly identical to each other.

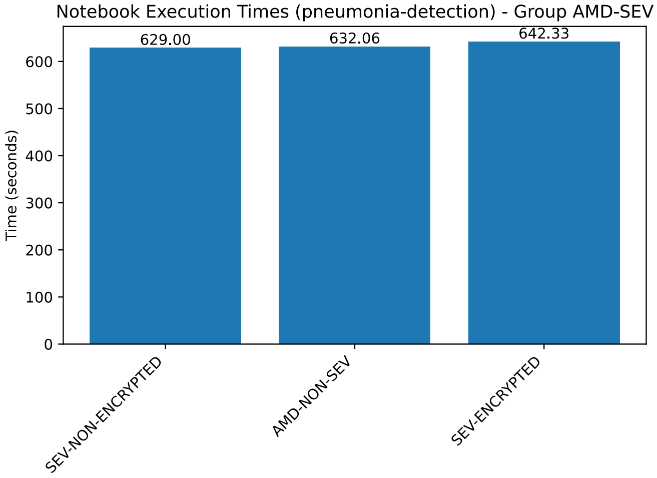

Figure 39: Pneumonia Detection Benchmark on AMD Systems

This notebook shows all AMD Machines having roughly the same result with no significant difference.

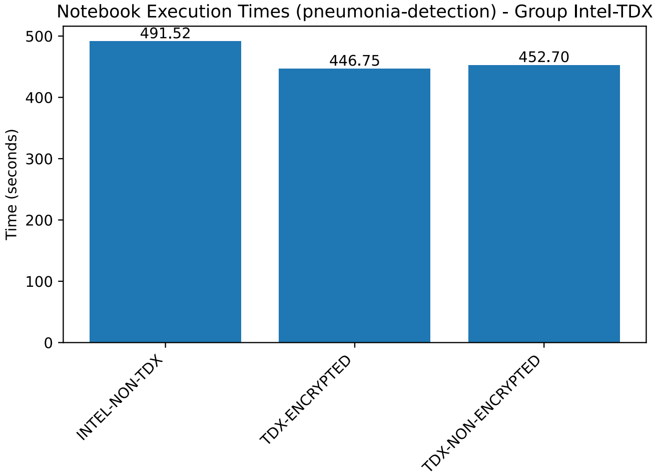

Figure 40: Pneumonia Detection Benchmark on Intel Systems

This notebook shows the non-TDX Machine having about 90% of the TDX ones while the TDX ones perform nearly identical to each other, making this a significant improvement for the non-TDX Machine.

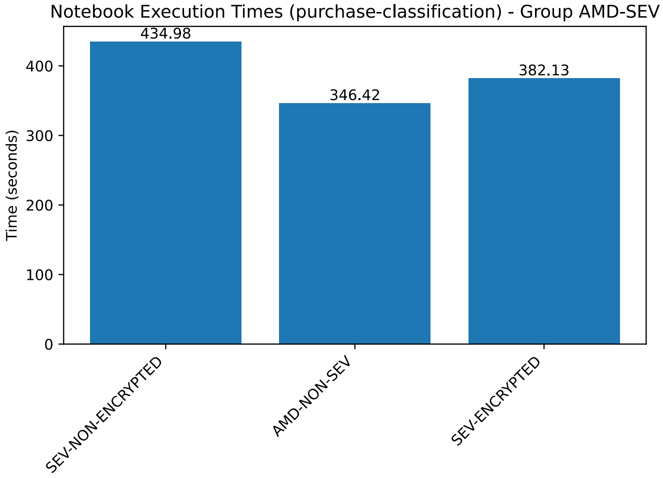

Figure 41: Purchase Classification Benchmark on AMD Systems

In this benchmark, the non-SEV machine is the fastest one, followed by the encrypted and finally the non-encrypted one.

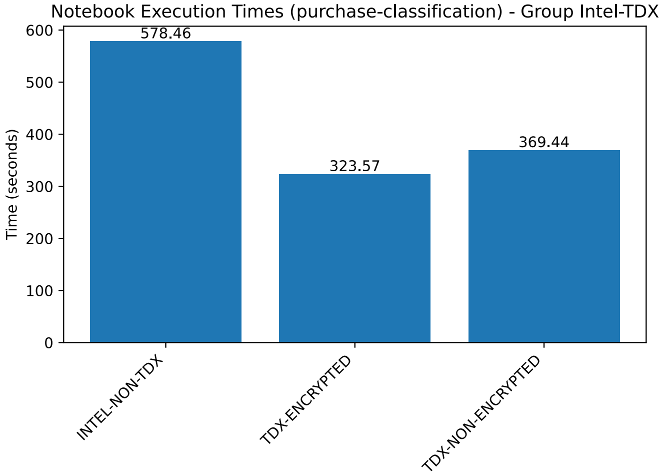

Figure 42: Purchase Classification Benchmark on Intel Systems

This notebook shows the non-TDX Machine having about 60% (578 vs. 3̃50) of the TDX ones while the TDX non encrypted one is slightly faster, making this a significantly worse result for the non-TDX Machine.

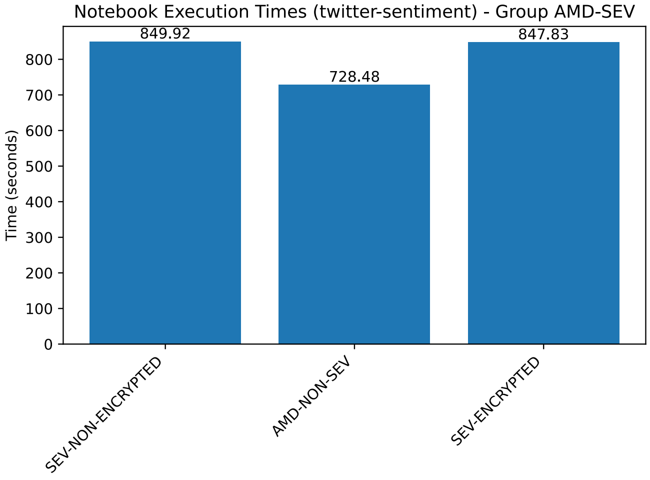

Figure 43: Twitter Sentiment Analysis Benchmark on AMD Systems

In this benchmark, the non-SEV machine is the fastest one, followed by the encrypted and the non-encrypted one.

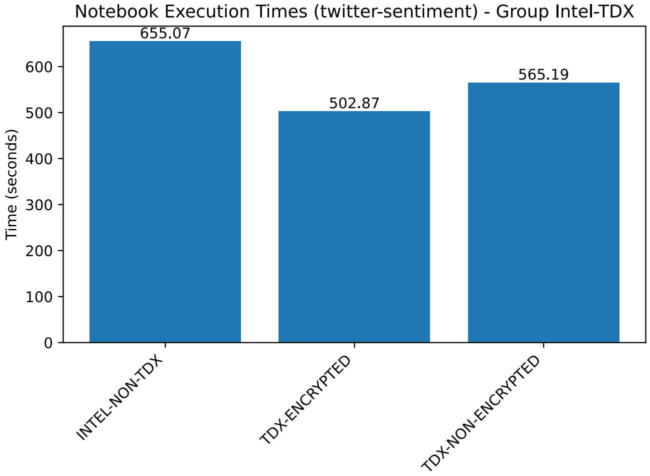

Figure 44: Twitter Sentiment Analysis Benchmark on Intel Systems

This notebook shows the non-TDX Machine having about 80% (655 vs. 5̃30) of the TDX ones while the TDX encrypted one slightly faster, making this an improved result for the non-TDX Machine.

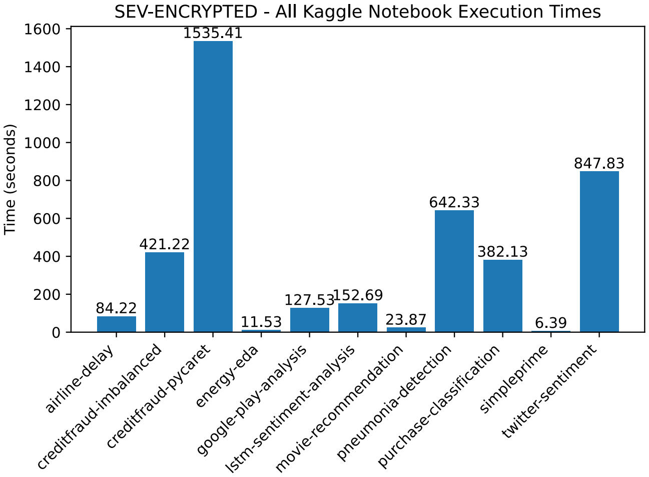

Figure 45: All Notebooks on SEV ENCRYPTED Machine

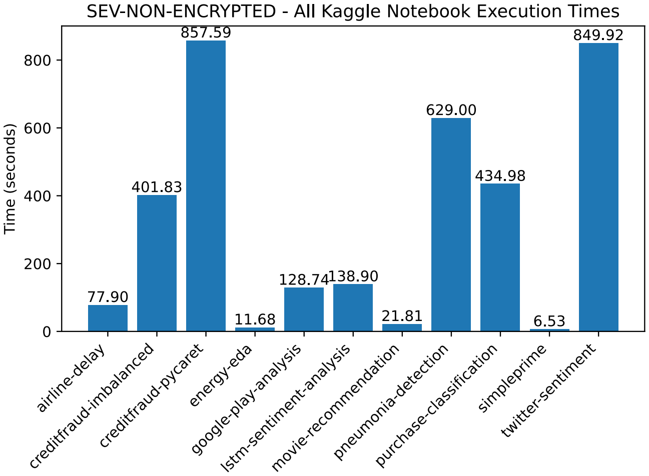

Figure 46: All Notebooks on SEV NON ENCRYPTED Machine

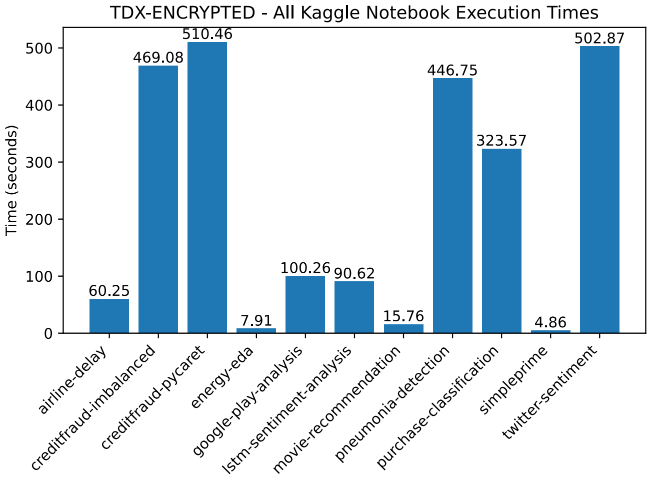

Figure 47: All Notebooks on TDX ENCRYPTED Machine

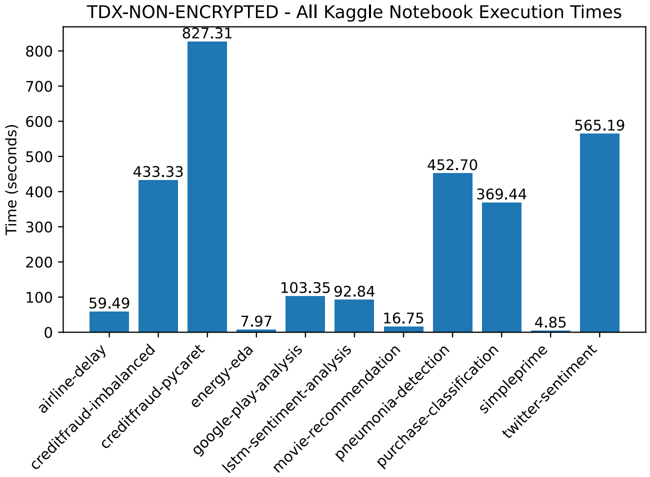

Figure 48: All Notebooks on TDX NON ENCRYPTED Machine

7.3.1 Kaggle Summary Notebooks.

8 CONCLUSION

In this section, we will highlight our findings and come to a conclusion.

8.1 Validation of the Kaggle Notebooks

We can see in figure 23 and 24 that the Simpleprime test performs as expected, and the variance is very small. This confirms, that the execution method can produce results with very little variance and high accuracy. This is good, because without the execution method being consistent, there would be no point in running the Kaggle notebook benchmarks.

8.2 General Findings

Generally we can see that one of the notebooks, the pycaret one, delivers very unreliable results. This shows in figure 29 & 30 where machines that should perform nearly identical, have huge performance discrepancies, amounting to about double the time for one machine (SEV-Enctypted vs SEV-Non-Encrypted) or (TDX-Enctypted vs TDX-Non-Encrypted). This is likely caused by the machine learning algorithm or implementation inside the notebook, making the run to run variance very high. As such, this notebook should not be considered when drawing a conclusion about the performance of a specific machine. Since we already confirmed the accuracy of our execution method with the simpleprime notebook, we can also rule out the execution method as our source of variance.

8.2.1 Enabling Encryption causes write latency. In all of our benchmarks, regardless of technology (TDX vs SEV) latency was higher for the writing of data shown in figure 17–22.

8.2.2 Read and Write Performance is largely unaffected. In contrast to the latency of write operations, most of the read and write operations show no difference in terms of IOPS or throughput. 12, 13

8.2.3 Not all benchmarks perform the same. Depending on the workload you put a System under, the performance difference could be huge. For Intel, all the PyPerformance benchmarks which mostly relied on CPU compute showed that the non-TDX machine showed about 75% of the performance of other machines in those tests, while in some tests like the creditfraud imbalanced notebook 28 the non-TDX machine was as fast as the other TDX machines.

8.3 AMD/SEV

While AMD would like to believe that the performance impact of enabling SEV is negligible, our performance tests show that this is not always true. While the CPU-heavy PyPerformance benchmarks showed no huge deviation from each other (3, 5, 7, 9) depending on the type of workload, the difference can be very large in our most extreme example (27) this performance difference was about 22% and up to 25% when disk encryption was additionally enabled. But this is not all of our results. Out of 10 Kaggle notebooks, 6 showed the pattern that enabling SEV is costing performance (43, 35, 41, 37, 27, 25), resulting in a longer execution time, with disk encryptions exaggerating this effect. While, 4 showed that the encrypted Machine was slower than the non-encrypted one. But this is not enough data to draw a conclusion, since there were some benchmarks where the encrypted machine was faster than the non-encrpyted one. To sum up: we could not without any doubt detect any performance impact of enabling disk encryption on SEV, but we could detect a negative performance impact between the non-SEV and SEV Machine when SEV was active.

8.4 Intel/TDX

The performance impact on Intel/TDX machines was different. Taking the roughly 75% performance of the non-TDX machine in the PyPerformance benchmarks as a baseline, the non-TDX machine performed about as expected in the majority of the notebooks. Sometimes scoring higher than expected with up to almost the same performance (28) down to only about 60% of the performance (42). While most of the benchmarks showed the non-TDX machine with about 75% of the TDX-Machine performance. Also, unlike with AMD, enabling disk encryption showed performance stayed largely the same, increasing and decreasing in a number of cases (44, 42, 28). To sum up: we could not without any doubt detect any performance impact of enabling TDX or enabling disk encryption on TDX.

8.5 Future Works/Improvements

Some notebooks like the pycaret creditfraud notebook could be exchanged with another notebook, which features less run-to-run variance. Or could be run multiple times to calculate a mean, although this is pretty performance intensive, with a run going as long as 1000 seconds. Because of the large variance and the pretty long execution time, the notebook should probably be removed. Executing more than 3 of the notebooks more than once could increase the stability of the results and decrease variance in the data. Because of resource and time constraints, this was not done in this work and could be improved upon in future works while maybe focusing on a comparison between 2 very specific machines, making it easier to compare and run an extended benchmark.

REFERENCE ENTRIES

[1] AMD. [n. d.] Confidential computing performance with amd sev-snp. Retrieved Jan. 1, 2024 from https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/performance-briefs/confidential-computing-performance-sev-snp-google-n2d-instances.pdf.

[2] Jens Axboe. [n. d.] Fio - flexible i/o tester. Retrieved Sept. 23, 2024 from https://fio.readthedocs.io/en/latest/.

[3] Yadu N. Babuji, Kyle Chard, Aaron Gerow, and Eamon Duede. [n. d.] Cloud kotta: enabling secure and scalable data analytics in the cloud. Retrieved Oct. 10, 2016 from https://arxiv.org/abs/1610.03108.

[4] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. [n. d.] Intel tdx demystified: a top-down approach. Retrieved Mar. 27, 2023 from https://arxiv.org/abs/2303.15540.

[5] Intal. [n. d.] Performance considerations of hardware-isolated partitioned vms with intel® trust domain extensions (intel® tdx). Retrieved Oct. 1, 2023 from https://www.intel.com/content/www/us/en/developer/articles/technical/tdx-performance-isolated-partitioned-vms.html.

[6] Intal. [n. d.] Performance considerations of intel® trust domain extensions on 4th generation intel® xeon® scalable processors. Retrieved June 9, 2023 from https://www.intel.com/content/www/us/en/developer/articles/technical/trust-domain-extensions-on-4th-gen-xeon-processors.html.

[7] Intel. [n. d.] Intel device attestation model in confidential computing environment. Retrieved Feb. 1, 2023 from https://www.intel.de/content/www/de/de/content-details/742533/device-attestation-model-in-confidential-computing-environment.html.

[8] Paul-Philipp Kuschy. [n. d.] Performance impact of disk encryption using luks. Retrieved Aug. 15, 2024 from https://scs.community/2023/02/24/impact-of-disk-encryption/.

[9] Charles Tappert Manoj Muniswamaiah Tilak Agerwala. [n. d.] Big data in cloud computing review and opportunities. Retrieved Dec. 17, 2019 from https://arxiv.org/abs/1912.10821.

[10] Jacob McPadden et al. [n. d.] A scalable data science platform for healthcare and precision medicine research. Retrieved Aug. 14, 2018 from https://arxiv.org/abs/1808.04849.

[11] Microsoft. [n. d.] About azure confidential vms. Retrieved Sept. 19, 2024 from https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview.

[12] Microsoft. [n. d.] Dasv5 sizes series. Retrieved Aug. 22, 2024 from https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/general-purpose/dasv5-series?tabs=sizebasic.

[13] Microsoft. [n. d.] Dcasv5 sizes series. Retrieved Aug. 22, 2024 from https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/general-purpose/dcasv5-series?tabs=sizebasic.

[14] Microsoft. [n. d.] Dcesv5 sizes series. Retrieved Aug. 22, 2024 from https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/general-purpose/dcesv5-series?tabs=sizebasic.

[15] Microsoft. [n. d.] Dv5 sizes series. Retrieved Aug. 22, 2024 from https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/general-purpose/dv5-series?tabs=sizebasic.

[16] Vikram Narayanan et al. [n. d.] Remote attestation of sev-snp confidential vms using e-vtpms. Retrieved Mar. 29, 2023 from https://arxiv.org/abs/2303.16463.

[17] Gamers Nexus. [n. d.] "not enough samples" - benchmark of cpu sample size ft. 68 cpus, hub, & der8auer. Retrieved Sept. 1, 2024 from https://www.youtube.com/watch?v=PUeZQ3pky-w.

[18] Victor Stinner. [n. d.] Benchmarks — python performance benchmark suite 1.0.6 documentation. Retrieved Sept. 22, 2024 from https://pyperformance.readthedocs.io/benchmarks.html#available-benchmarks.

[19] Victor Stinner. [n. d.] The python performance benchmark suite. Retrieved Sept. 22, 2024 from https://pyperformance.readthedocs.io/.

[20] Linus Tech Tips. [n. d.] Amd says these are the same... we disagree. - testing 12 of the same cpus for variance. Retrieved Sept. 1, 2024 from https://www.youtube.com/watch?v=os-jXiYRihI.

[21] Qifan Wang and David Oswald. [n. d.] Confidential computing on heterogeneous cpu-gpu systems: survey and future directions. Retrieved Aug. 21, 2024 from https://arxiv.org/abs/2408.11601.