CSC8199 : Performance Evaluation and Comparison of Trusted Execution Environments for Microservice Applications

Ronil Rodrigues
230112209

MSc in Cloud Computing,
School of Computing, University of Newcastle, U. K.
r.rodrigues2@newcastle.ac.uk

Abstract. Performance and security of an application are important factors when considering a secure execution environment. Hardware assisted trusted execution environment like Intel Trusted Domain Extensions and AMD Secure Encrypted Virtualization provides such secure execution environment by isolating and encrypting the environment. The research here conducts a severe benchmarking on cloud hosted machines by running multiple micro and macro benchmarks to evaluate the performance of the applications when utilizing the trusted execution environment features. Different benchmarking strategies were employed for this research which includes benchmarking high performance computing application using y-cruncher, mysql microservice application and other micro benchmarks like sysbench and netperf. The research concluded that even though trusted execution environment provides an added security benefits, their impact on performance of an application varied depending upon the workload. This study aims to provide a framework helping organization decide a suitable trusted execution environment based on their performance for a particular application.

Declaration: I declare that this dissertation represents my own work except where otherwise explicitly stated.

1 Introduction

With rapid growth in cloud computing and the number of organizations moving their workload to cloud, ensuring the security and confidentiality of sensitive workloads has become critical for the businesses. Various technique have been employed to protect the workloads in cloud which includes protecting the data using various firewall and software which is not always reliable and comes with added performance overheads [1]. Trusted Execution Environment is a new way of protecting such cloud systems where sensitive data is processed [2]. Trusted Execution Environment offers a viable solution in the form of isolated environments that protect data and applications from potentially comprised host systems and other hostile entities [3]. Many prominent processor manufacturers have created their own trusted execution environment to provide a better protection to the workloads in cloud, which include AMD SEV SNP and Intel TDX.

Our current study focuses on deploying confidential virtual machines to assess the impact of different Trusted Execution Environments on the performance and efficiency of different micro-service applications [4]. By utilising cloud infrastructure, we can collectively benchmark and compare the performance characteristics of Intel TDX and AMD SEV SNP. This performance evaluation includes comparing key metrics such as latency, resource utilization, throughput and scalability to determine how each Trusted Execution Environments influences the operational efficiency of micro-service applications.

This research is motivated by the need to address the growing concerns around data security and privacy in cloud computing. Cloud providers, including Microsoft Azure, Amazon Web Services (AWS), Google Cloud Platform (GCP), are increasingly adopting policies such as zero trust to enhance their security frameworks [5]. By conducting rigorous performance evaluations, this study aims to help organizations in selecting the most appropriate Trusted Execution Environments for their specific use cases, thereby ensuring that their confidential workload remains secure and compliant with the highest standards of data protection. As the tech landscape continues to change, security challenges and attacks increases which leads to the risk of managing any sensitive workloads in cloud, so this research will provide companies with a reference in selecting a trusted execution environments that can protect their system without compromising on performance.

1.1 Aim

The main aim of this study is to analyze and compare the performance of different Trusted Execution Environments (TEE), such as Intel TDX and AMD SEV SNP, by deploying confidential virtual machines on Microsoft Azure Cloud and analyzing their impact on the functionality and efficiency of micro-service applications.

1.2 Objectives

Understanding the implementation of Intel TDX and AMD SEV SNP : Doing a background research on Intel TDX and AMD SEV SNP to understand the need of the infrastructure and how the hardware helps protects the user data and applications. Understanding the overall architecture in place to see how the code works when executed through the hardware. Researching on the current use-cases in the industry.

Running Applications in Local Virtual environment : Utilising the Occlum Api to test different micro service applications provided by occlum in simulated SGX virtual machines.

Learning about confidential VMs and implementation : Researching about the configuration requirements for confidential virtual machines and how applications can be made to run under the secure environment.

Creating Microsoft Azure VMs and trying to implement applications on Public Cloud : Creating different confidential virtual machine for AMD SEV configuration and TDX configuration. Setting up the environment by switching to secure computing mode to run micro service applications.

Running Micro Service applications on Public Cloud : Once the initial setup is done, our objective would be to run micro service applications like Mysql and High Performance Computing application to check the feasibility of the environments.

Automating the Pipeline : Creating a unified script to automate the whole benchmarking process for each micro and macro benchmarking applications.

Performance Analysis of Micro Service Applications on different Trusted Execution Environment : Analyzing different metrics and comparing the performance of different trusted execution environment by running similar micro services applications. Suggesting the optimal TEE in terms of security and performance.

1.3 Document Structure

The structure of this dissertation follows the following format:

- Background. This section expand on the Trusted Execution Environment technology, the types of TEEs, their working and their evolution.
- Methodology. This section focuses on different benchmarks employed, working and execution.

- Results and evaluation. This section looks at the performance of application when employing different benchmarking strategy and provides an analysis on the same.
- Conclusions. This section summarizes the work completed, revisits the aim and objectives, and suggests future work which could help companies select the best TEE.

2 Background

This Section reviews background and why the need of a new secure environment was required. The research and implementation proposed here helped in the implementation process. We do analysis of the working of trusted environment and how confidential computing plays a big role in its implementation on cloud. The section also emphasizes on the need of such technology and how it's been implemented in current tech landscape.

2.1 Trusted Execution Environments

As the systems are evolving and have been made public, challenges for security specialists to protect users from potential attacks have been increasing drastically [6]. Attacks have also evolved and are more complex and unexpected, that leads to increase demand on protection tools and architectures to safeguard the systems. The traditional security systems and architectures are not capable of dealing with changing complex and connected systems which resulted in introduction of Trusted Execution Environment. To develop a system on Trusted Execution Environment it was important to study the need of using such environments. Trusted Environments refers to the use of separate hardware module to provide functional interfaces to protect and secure execution platforms for computer systems [7]. Trusted Environments also provides a safe execution environment for the code to execute, which in turns provides a secure environment for the application to run. Trusted Execution Environment (TEE) are hardware based solutions designed to create a secure execution environment that is isolated from operating systems and ensure that sensitive that is protected from any unauthorized access or attacks. The protection of sensitive data is critical because current cryptographic methods fail to secure data during processing, exposing the data to potential attacks and threats [8].

Fig.1 gives a general structure of Trusted Execution Environment. Any Computing environment generally consist of one software and one hardware layer. The software layer of TEE consists of one side which includes operating system and applications referred to as untrusted area. The other side of TEE is referred to as trusted area, which will be utilized for running applications securely. Both the execution environment exist concurrently in the same computing environment, having their own separate software stack. Open Mobile Terminal Platform (OMTP) in [9] refers to TEE as an environment that provides isolation to an application protecting it from different type of attacks. It is recommended for user

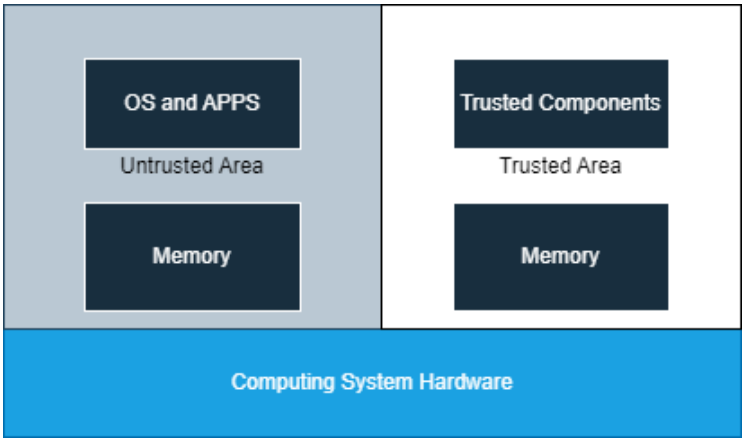

Fig. 1. Trusted Execution Environment

application to run in TEE when handling sensitive data like user information, password, etc.

TEE are broadly classified as hardware encrypted TEE and software encrypted TEE, we will have an overview of the both classifications before moving into our main selected TEE for this research.

2.2 Software Implemented Trusted Execution Environment

A Software Implemented trusted execution environment provides a more secure space inside the main processor which ensure that data is stored securely and processed without the risk of breaches. A Software Implemented trusted execution environment provides a more practical and flexible way to secure data where hardware implemented trusted execution environment cannot be used. Software Implemented TEE can be deployed on a standard hardware to get features of hardware implemented TEE without security features. Software Implemented TEE creates a secure enclave within the system main processing unit. The newly created enclave will isolate the application processing from rest of the system which help in securing the application. Software Implemented TEE utilizes system features like secure context switching, memory protection keys and other technologies to create an isolated execution space for an application to run. TEE is initialized during the booting of the system which ensure the integrity of the other software components. Once implemented the TEE can run applications in complete isolation from the operating system and other applications which prevents unauthorized access and also protect the overall applications. The application data handled by the TEE is encrypted in transit as well as rest, using cryptographic encryption techniques like AES to protect it from tampering. TEE also provides remote attestation which allows external application to verify the authenticity and integrity of the environment [10].

In real life application, Software Implemented trusted execution environment are important to ensure privacy and data security in multi tenant environments. Software Implemented trusted execution environment has a number of benefits over the hardware implemented trusted execution environment like it is not dependent on any special hardware feature and can be updated easily without any added cost. The fast update features helps to tackle any vulnerabilities found at a later stage of application implementation. Another benefit of using software implemented TEE is it does not expand into hardware based TEE, which helps in using them both concurrently [11]. Few examples of software implemented trusted execution environment are Microsoft Virtualization-Based Security (VBS), Open Enclave SDK, IBM Secure Service Container. IBM Secure Service Container provides a more secure environment for executing containerized applications. It ensure isolation of container runtime and protecting data by using software based TEE. All of these software implemented TEEs may vary in their implementation and architecture but the end product is same which provides isolated execution environment to make our applications more secure. Software Implemented trusted execution environment helps enterprises to implement trusted execution environment without depending upon the underlying hardware providing scalability and flexibility.

2.3 Hardware Implemented Trusted Execution Environment

Hardware Implemented Trusted Execution Environment is a secure part of the main processor which provides isolation to the application which helps in making the application and runtime more secure. As per [12] Hardware based Trusted Execution environment uses some hardware backed technique which helps in increasing security for the execution of code and protection of data within the secure environment. Hardware Implemented TEE must have the following security properties:

- It should provide a measurement of the hardware platform and provide information about initial state of the TEE instance.
- It should prevent any unauthorized access to the code and data.
- It should keep the internal data confidential against software and other physical hardware.

The Architecture of an Hardware Implemented Trusted Execution Environment consist of numerous key component that help in creating a secure execution environment for the code to run safely which are described below.

Processor Extensions : Many of the TEE available in the market incorporate specific extension which provides a set of instructions that allow applications to create secure enclaves within the process that helps in creation and management of Trusted Execution Environment. Intel SGX can be considered as one example where the processor allows creation of enclaves that isolates applications and

code [13]. Enclaves can not be accessed by even operating system and hypervisor. Processor Extensions play a crucial part in providing a secure execution environment.

Memory Encryption : Memory Encryption is another important component of Trusted Execution Environment that encrypt the data stored in the system memory thereby protecting the applications data from attackers as they can't read or modify the data even if they get access to the physical memory hardware. Memory Encryption provides a more secure option to handle critical components like cryptographic keys, personal data by encrypting data before it is being stored in memory by a secure key and decrypting the data whenever required. This approach helps reduce physical attacks like direct memory access attacks and also cold boot attacks by encrypting the data protecting them from being compromised. The hardware encryption encryption provides and extra layer of protecting thereby minimizing the risk of vulnerabilities. The key utilized for encryption are never exposed and are managed by hardware.

Secure Boot : Secure boot help in initializing TEE in a trusted state. The process help in preventing malicious code from being executed during the initial state of system startup which helps in securing the environment [14]. The booting process involves a series of cryptographic checks verifying each stage before being executed. The firmware, bootloader and operating system kernel are all verified using digital signatures, which ensure that only trusted code can be executed. If any error occurs or any component fails the verification process the whole boot process is halted which help in preventing any further problems.

Runtime Isolation : The runtime isolation is a very critical features of the Trusted Execution Environment as it provides one complete separation executing environment for the sensitive operations from other codes and applications [6]. The integrity and security of the code and data of the software module can be enhanced by using such isolation practices. The isolation is enforced by the hardware, which ensure that the application operates without the interference of other applications and operating systems. This ensure that even if main operating system is compromised by any attacks, the trusted execution environment remains secure. Intel SGX TEE creates enclaves which run in a separate memory space which can not be accessible by operating system enforced by the hardware.

Remote Attestation : Remote Attestation is a feature where a verifiable report is generated detailing the initial state of the system to verify authenticity and integrity of the code and application [15]. The initial states of the TEE consist of a cryptographic hash of the application as well as the TEE hardware configuration which can be verified by a remote user. Most TEEs follow similar procedure to generate attestation report [16]. In Intel SGX, enclave generates an attestation report that includes hash of the initial state and other configuration

hardware information which is signed by a trusted attestation service, which provide assurance that enclave is safe and secured to any remote party involved. Remote Attestation is a key part in building trust among remote server where sensitive information is involved.

2.4 Intel TDX

2.4.1 Evolution of Intel TDX

Intel TXT (Trusted execution technology) is used to create a stable environment from the start of the system by using specific Intel CPU, dedicated hardware and related firmware which provides a system with multiple methods to achieve a safer execution environment. Intel entered the world of Trusted Execution Environment in 2015 with Software Guard Extension based on Intel TXT which provided it with capabilities to create secure enclaves to process sensitive data. Intel SGX introduced a hardware based security encryption to isolate application/code execution with secure enclaves protected from the other systems and applications. Intel SGX helped organisations to build applications that could process and securely handle confidential data even if operating system and hypervisor are compromised. The isolation provided by Intel SGX inside the enclaves made sure that the data is encrypted and is secure which assured data integrity and confidentiality. Intel SGX enabled many companies across different sectors an alternative to keep their sensitive data more secure [17].

Fig. 2. Evolution of Intel TDX

As shown in Fig.2 in 2017 Intel came with the next version of Intel SGX known as SGX 2.0 which had all the features of SGX 1.0 and some added features like dynamic memory management inside the enclaves, which allowed applications to allocate and deallocate memory as per requirement boosting performance of the applications and also providing flexibility to the application. With rapid innovations and success of Intel SGX, Intel introduced a newer version of trusted execution environment with Intel TDX also known as Trusted Domain Extension. Intel TDX was built on top of Intel SGX enclave capabilities with added functionalities enabling secure data sharing between multiple hosts. The Integration provided by Intel TDX with underlying SGX helped in meeting security criteria for evolving cloud environments and applications. Intel is still working on developing new hardware encrypted TEE by utilizing the capabil-

ities of Intel SGX and Intel TDX to accomodate need of multi tenant cloud environment for evolving security needs of organizations.

The introduction of Intel SGX and Intel TDX has revolutionized Intel's commitment to address challenges of cyber attacks and data protection. These technologies are helping Intel create a secure and trusted execution environment and also aided the adoption of secure cloud and multi tenant computing environments.

2.4.2 Intel SGX

Intel Software Guard Extensions (SGX) is a set of hardware extensions to the Intel architecture that aims to provide integrity and confidentiality sensitive computation performed on a computer where kernel, hypervisor are considered to be potentially malicious for the functioning of an application [18]. Intel SGX allows application to run in secure containers inside of them called as enclaves with dedicated memory regions that are secured with memory encryption where access to the encrypted system memory is managed by the hardware excluding operating system or any other application [13]. If any untrusted code tries to access enclave memory, SGX disables the environment and denies entry and prevents any change in the content of application [19].

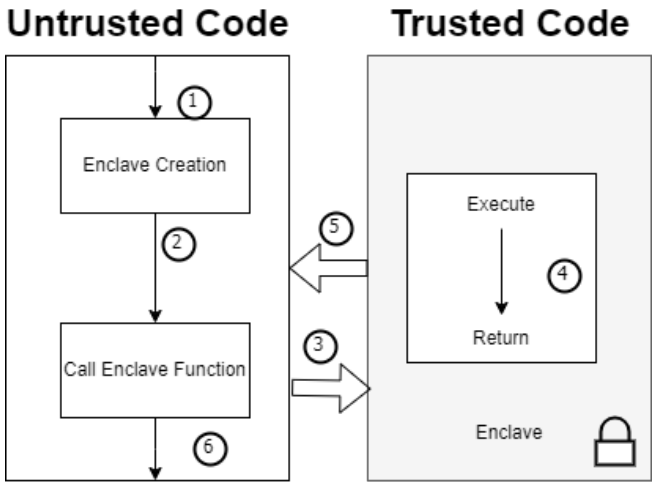

Fig. 3. Intel SGX Architecture

Intel SGX uses two main features that help in securing applications i.e Isolation and Attestation. Software Attestation is a feature that allows any remote party to authenticate application running inside enclaves. Isolation is a feature that isolates enclave environment from any untrusted application interview running outside the enclaves.

As per [20] remote Attestation helps a remote party confirm that an software is running inside a secure SGX enclave. Remote Attestation can be performed when the enclave are initialized during the whole lifecycle of an application. In reference to Fig.3 When an SGX enclave is initialized, a cryptographic hash of the code and data in enclave is created, this serves as an unique identifier which is stored in the encrypted stage. The key used for the encryption is unique to the enclave based on the system specification. Intel SGX has a special enclave called as quoting enclave designed to handle remote attestation using asymmetric cryptography. The quoting enclave creates a quote that includes system measurement and a report created by the hardware which is signed using Intel provided encryption key. The remote party can receive the signed quote and verify the system using Intel Attestation Service which checks the quote against a wide range of measurements and configurations. Once the verification step is done, the remote party can conclude on deciding whether to trust the enclave which contains the application code. Once verified the remote party communicates with the enclave using secure channel which is established during the attestation process. Remote attestation in all together ensures that data is protected within and enclave and also can be verified by any other external third party, which makes it an important tool to use in evolving cloud computing world.

Even though Intel SGX provides a secure confidential environment but it suffered performance degradation because of various memory issues due to enclave context switching and enclave migration. As per [21] the main issue and limitation are related because of the added security mechanims used to secure the enclaves. Since the data stored in enclave is encrypted in the main memory there is an added computational cost as data travel between memory and CPU. Absence of a trusted path to exchange sensitive data with different input output devices is also an important parameter to consider while using Intel SGX.

Intel SGX is also prone to a number of side channel attacks like cache attacks which can cause leak of data from the encrypted SGX enclaves and also microarchitectural attacks that can find vulnerabilities inside your code. The Intel SGX enclave architecture uses the Intel TXT to bring a new concept that aimed to ensure the confidentiality of applications and code providing a trusted execution environment to protect sensitive applications to tackle security challenges but it should be considered that only data inside enclaves was protected which led to the development of Intel TDX.

2.4.3 Intel TDX

Intel SGX focused on creating secure enclaves for an application to run and perform operations in secure manner, which had drawbacks since only enclaves part of the system was encrypted, so Intel came up with another version of trusted Execution Environment known as Intel Trusted Domain Extensions i.e Intel TDX where instead of encrypting just an enclave whole virtual machine was encrypted. Encrypting the whole virtual machine protects applications from various threats which include compromised hypervisor and other cyber attacks.

Intel TDX encrypt data at rest and in transit within the system memory which ensures that the data of the virtual machine is not compromised. One major improvement of Intel TDX over Intel SGX is the enhanced isolation which ensures that compromised hypervisor cannot get access to the code inside the virtual machine providing isolation to sensitive workloads [22]. Intel TDX has a secure management of key which are stored in hardware protection them from being exposed. Intel TDX is built on intel SGX but with a better remote attestation mechanism , which allows remote party to verify whether a TDX protected virtual machine can be trusted or not [23].

Intel TDX secure virtual machine (VM) by using Intel multi key total memory encryption and CPU attested software module by providing complete isolation. Intel TDX module is digitally signed module that is executed by a processor mode known as Secure Arbitration Mode [24]. The CPU in the Arbitration Mode lacks full permissions, which restrict it from accessing other secured memory regions, such as regions used by Intel SGX enclaves or any other CPU system management mode. Installation of this TDX module is generally taken cared by the SEAM Loader as shown in Fig.4 which is an authenticated code module integrated into Intel Trusted Execution Technology Stack. It is responsible to verify and load the Intel TDX module into SEAM memory range after being invoked by Virtual Machine [25]. The SEAM Loader passes the security version number through hardware registers and places TDX module in SEAM VMX root mode [26]. Once the module is initialized, the virtual machine can communicate and pass control to TDX module by executing SEAMCALL instructions sequentially as shown in Fig.4. SEAMRET instruction is generally used by TDX module to return execution control to hypervisor and ensuring operations like creation, deleting of a Trusted Domain (TD). VMLAUNCH and VMRESUME commands are used to start or resume vm, this command moves TDX module into SEAMVMX non root command mode before passing control to TD [26].

As discussed in [27] Intel TDX utilizes VMX to enforce memory isolation. Intel TDX encrypted applications are unable to access any other domains like hypervisor or any other virtual machines trusted zones. In case of Intel TDX as hypervisor are considered to be non trusted entities, Intel TDX has it's memory managed specifically by TDX module which also controls the address translation of Trusted Domain (TD) private memory. TDX memory is protected from privileged software, malicious applications by implementing access control and cryptographic isolation. Cryptographic isolation protects against malicious software from accessing memory or corrupting TD memory. In TDX enabled Intel machine, physical memory is partitioned into normal memory and secure memory where sensitive data of TD should be stored. TDs can also allocate some memory as shared memory for I/O which is not under protection from encrypted TD and thus is stored in normal memory. Software which are not executing in SEAM mode generally belong to the normal memory with no access to secure memory. Memory access checks are enforced by memory controller [27]. An additional memory pages can be added to secure memory if requested by hypervisor, the TDX module then keep it's state in the PAMTs (Physical Address Meta

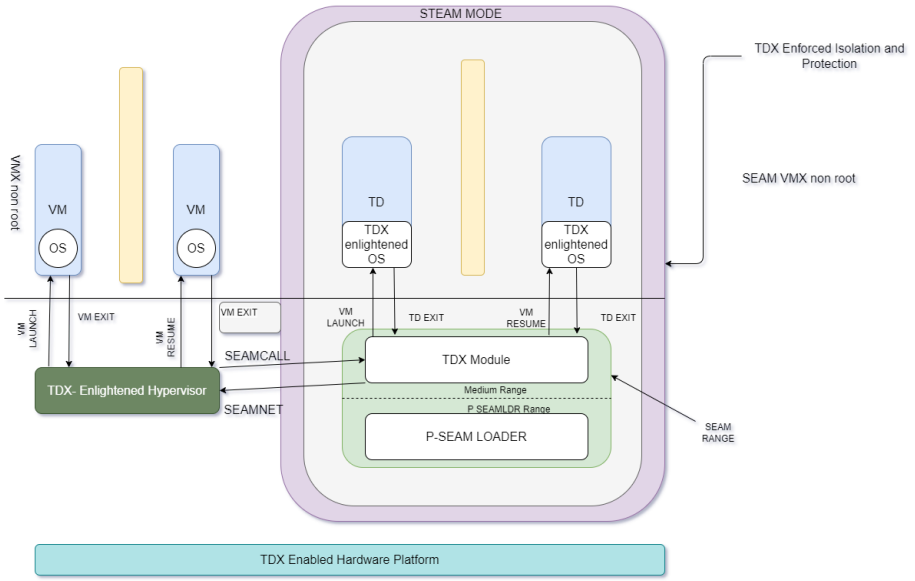

Fig. 4. Intel TDX Architecture

Data Table) data structure and zeroes the corresponding memory page. A virtual machine sets up first stage memory translation by utilizing secure memory page table. The highest bit of intermediate addresses generally identifies secure memory. Configuration of second stage memory is handled by the TDX module using PAMTs to restrict multiple owners from using same physical page providing isolation to the virtual machine. Accessing a secure page sets SEC bits of ECC memory, which is useful for access management during context switching making the memory inaccessible to hypervisor. Intel TDX uses Multi key total memory encryption (MKTME) for encrypting TD's private memory, which is also responsible for transparent memory encryption and decryption of the data passing through memory controller. During the creation of virtual machine MKTME assign unique keys to encrypt data passing to the memory. The keys are associated with the Host Key identifier (HKIDS) in the physical address which is used by MKTME to perform cryptogrpahic operations with keys stored in its internal memory [27]. Intel TDX comes with two distinct mechanisms for ensuring memory integrity that includes Logical and Cryptogrphic Integrity. Logical Integrity protects against software attacks by preventing unauthorized encryption and read of secure memory. It checks whether the processes running inside the SEAM mode can encrypt with private HKID and reach a cache lines with their initial TD owner bit set to 1. Intel TDX considers peripheral devices as untrusted and are restricted from accessing private memory of TDs. It is the responsibility of the TD to secure I/O data buffer and placing them in shared memory, which can be identified by the share bit in the guest physical address.

Hypervisor and Peripheral can then move data through the shared memory as required [27].

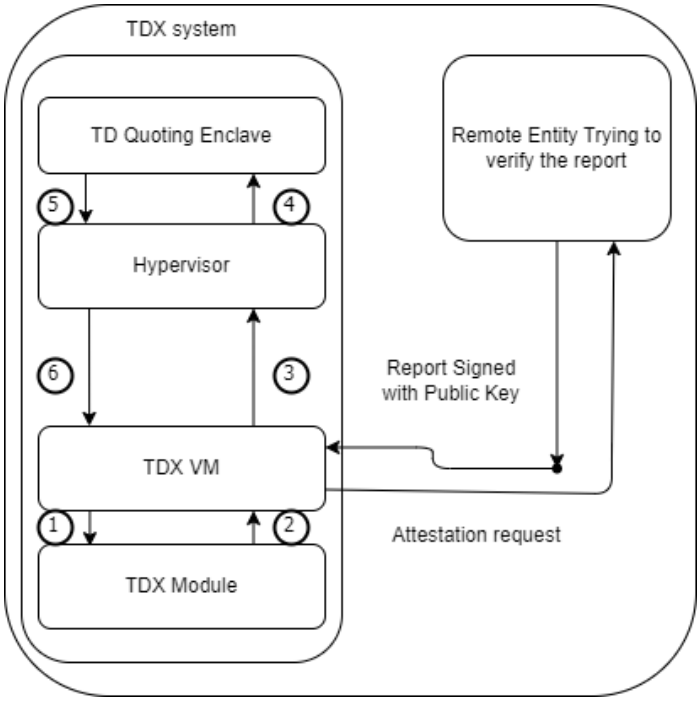

Fig. 5. Intel TDX Remote Attestation

Another important method for verifying identity of a Trusting Execution Environment is by using Remote Attestation. Verification is done by a remote entity to make sure that the initial state of the TEE is not changed and can be trusted. Fig.5 shows the whole remote attestation method employed by Intel TDX. In Intel TDX, attester is responsible for handling remote attestation requests which is situated within the TD. TD report is generated inside a TD that is used a proof of instantiation and proper functioning of the virtual machine. When a remote entity issue a request for verification, attester provides an evidence of proper instantiation of TD and some user data in a report. To enable verification of this report is need to be converted into a quote. Intel TDX utilizes the same features as SGX to perform remote attestation. This quote is generated by the TDX module and is then signed by the quoting enclave which contains by the measurements of the TDX TCB and software components that are loaded in the corresponding TD. The quote generated by quoting Enclave generally also includes a certificate chain anchored by Intel certificate [27]. This quote is then shared to the remote party for verification. Once the remote party success-

fully verifies the authenticity of the TDX Virtual machine a secure channel is established between attester and the remote party.

2.5 AMD SEV

2.5.1 Evolution of AMD SEV

As shown in Fig.10, AMD came up with their first version of Trusted Execution Environment in the year 2016 by introducing AMD Secure Encrypted Virtualization (SEV) an hardware based CPU extensions which provided encryption to protect sensitive workload from unauthorized access. AMD SEV is build on AMD secure virtual machines which is an instruction set for specifically hardware accelerated virtualization [28]. AMD SEV provides isolation to application running inside of them enhancing security. AMD SEV provides memory encryption to virtual machine memory which helps in preventing attacks on the VMs. Another key feature of AMD SEV was key management which was done by a secure processor, helping reduce risk of exposure. This key is stored in the AMD Secure Processor (SP), a separate security co-processor based on ARM Cortex A5, isolated from the main CPU. Upon the start of a virtual machine, the AMD SP generates and securely stores unique keys to encrypt all protected memory of the virtual machine. With the foundation of SEV, AMD came up with a new technology in 2018 known as SEV Encrypted State (SEV ES). AMD SEV ES inherited all the original features of AMD SEV along with few added features like CPU registers and enhanced isolation for the VM's. AMD SEV ES provides a enhanced isolation which made sure that the hypervisor was not able to access the data or tamper the data. AMD SEV ES also encrypts CPU register data which helps protects against malicious attacks.

Fig. 6. Evolution of AMD SEV

AMD kept improving the SEV versions and in the year 2020 they launched another TEE named as AMD SEV Secure Nested Paging (SNP) which builds upon previous extensions and provided better integrity guarantee by preventing unauthorized access and changes to the VMs memory. AMD SEV SNP also inculcated attestation feature that increased security of the VMs which can verify the configuration to prevent any attacks related to rollbacks. AMD SEV SNP also had some optional features like security enhancement to handle additional VM models, it also offered protection around different interrupt behaviour, it also protected against side channel attacks.

2.5.2 AMD SEV

AMD Secure Encrypted Virtualization is a technology introduced to tackle security related issues by encrypting VM and providing isolation to the application. AMD Secure Processor as shown in Fig.7 is the building block of AMD SEV architecture which is a focused security system within AMD processor. AMD SP performs encryption and also stores encryption keys which helps in maintaining integrity of AMD environment. AMD SEV encrypts VMs memory space with an encryption key thus protecting VM from compromised entities and hypervisors. AMD SEV is an crucial advancement as companies are moving to cloud where data security and privacy are important parameters for any company [29]. AMD SEV uses the same encryption engine as used by AMD Secure Memory Encryption where a dedicated c-bit is used in page table to mark which memory page is encrypted which ensure that data in those pages is automatically encrypted and decrypted by the controller.

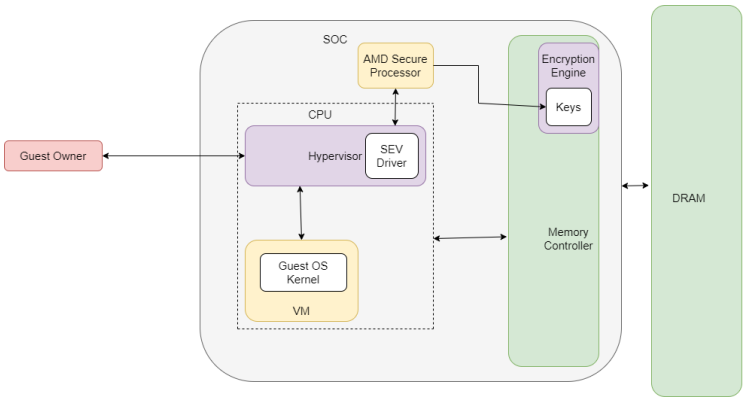

Fig. 7. AMD SEV Architecture

AMD SEV encrypts the memory and software running inside the SEV protected VM without any change to the host applications. AMD SEV utilizes AMD Memory Encryption Engine for encrypting and decrypting VM memory which creates an unique encryption key for each VM. During VM creation, hypervisor uses AMD SP to create the encrypted memory space encrypted with a VM specific key which restricts other entities from accessing the memory. The data that is being written to memory space is encrypted and the same data can be decrypted before being read, this whole lifecycle is performed by Memory Encryption Engine which ensure that data is always secure even during operations. When an application or code is written onto the AMD SEV encrypted memory space, SEV tags the code with a unique key that is distinct to just that particular encrypted memory space. As per [30] in AMD SEV guest VMs can choose which data pages need to be private which allows VMs authority to keep particular

data pages to be confidential and secure. Private memory is encrypted with a specific guest key and shared memory is encrypted with hypervisor key.

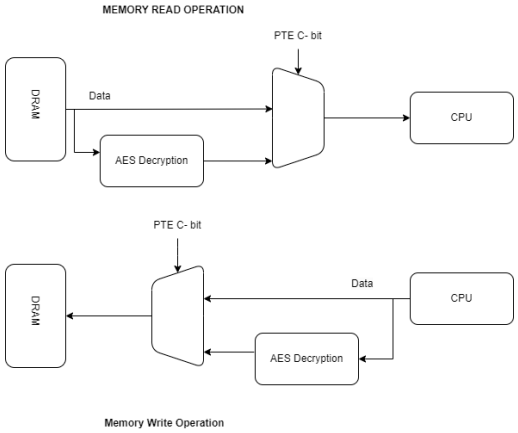

Fig. 8. AMD SEV Encryption

AMD SEV takes cares of security parameters but relies on hypervisor for functions like scheduling and emulation, VM running with SEV enabled mode uses hypervisor as per requirements but secures it's memory space when it's not shared. When an VM is launched in AMD SEV, AMD SP loads appropriate encryption key in AES 128 encryption engine. Even though hypervisor has access to the major components in the processor, it is not considered to be a security threat as the VM is already encrypted with a specific key.

2.5.3 AMD SEV ES

AMD SEV Encrypted State (ES) was build on top of features of AMD SEV, with enhanced security of VMs as AMD SEV ES extended encryption to include the whole CPU register state [31]. In AMD SEV architecture, storing and restoring VM register state is multi level process which includes VMRUN instruction to transfer control to a VM. In a typical guest, hypervisor has to load the guest information using VMLOAD instruction which involves a number of steps and instructions that reduces the performance and efficiency [32].

AMD SEV ES architecture combines all the operation into one simple hardware instruction i.e in VMRUN. when the instruction is loaded in AMD SEV ES the hardware loads all registers information and when VM stops, all information about the different state are saved automatically to memory and hypervisor state is again loaded, this state is memory encrypted with VMs memory encryption key which is not shared with any software and hypervisor cannot read the register state. AMD SEV ES architecture ensure that only the VMs content is secure and cannot be tampered by hypervisor. An integrity check value is used

to detect any modification to the VM state since last VMEXIT. VMEXIT event can be of two types Automatic Exits (AE) and Non Automatic Events (NAE). NAE events are triggered when guest VM requires hypervisor emulation while AE events occurs on shutdown or asynchronous interrupts. In AMD SEV ES, AE events are the default events which transfer control to the hypervisor once triggered. When the events are triggered CPU hardware saves and encrypts all register state and load the original hypervisor state. As shown in Fig. 9 NAE event triggers a new exception VC (Virtual Machine Communication Exception) which is handled by the VM. Decision to provide access to the hypervisor lies with the exception handler.

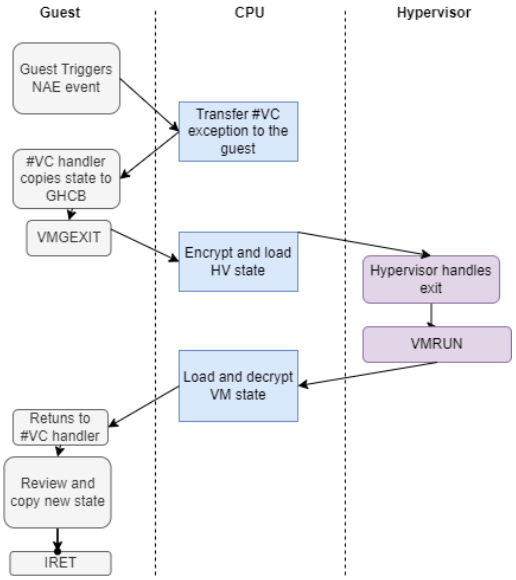

Fig. 9. Non Automatic Event Flow

A specific shared memory is utilized to exchange the data known as Guest Hypervisor Communication Block where values to be copied are stored, then VM executes VMEXIT which transfer control back to hypervisor [33]. In case of AE events like Nested page faults, hypervisor is not required to read VM state, so VC exception is not triggered. To handle this scenario hypervisor sets up a reserved bit in nested page table to enforce nested page faults as NAE events which results in VC exception being raised. Since in AMD SEV ES most of the instruction cracking like tasks are moved into VC handler, the need for hypervisor emulation support reduces significantly. VC exception provides additional features to optimized VM and hypervisor contact along with the primary duty to protect register state.

2.5.4 AMD SEV ES SNP

AMD SEV ES Secure Nested Paging (SNP) built on top of AMD SEV and AMD SEV ES extended the use case of TEE by providing remote attestation and migration functionalities. AMD SEV SNP has advanced features like memory integrity, cryptographic isolation and secure boot which ensures that state of VM is secured even though hypervisor can be compromised. AMD SEV SNP supports remote attestation which allows a remote user to verify status of VM which is an important feature when using cloud environment where security can be compromised.

Fig.10 describes remote attestation lifecycle for an AMD SEV SNP VM. A third party can request attestation report at any time to verify state of the VM. The communication between VM and AMD SP takes place with an encrypted channel. SEV firmware generates an attestation report which includes measurement of the initial VM state and also arbitrary data can be included in the report which is then provided to a third party like a remote entity [34]. Attestation report is signed by an platform endorsement key known as Versioned Chip Endorsement Key (VCEK). AMD Root key signs and verifies the VCEK which ensure the authenticity. Remote third party can verify the report utilizing the ARK and VCEK that authentic AMD SP is issuing report [33].

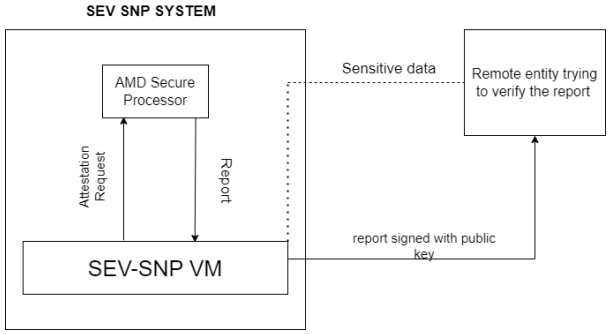

Fig. 10. Remote Attestation of AMD SEV SNP

AMD SP is used for migration of SEV protected VM in case of AMD SEV and SEV ES, but AMD SEV SNP introduced Migration Agents (MA) to handle the migration efficiently. In the initial step hypervisor utilizes the `SNP PAGE SWAP API` command to migrate a VMs memory. AMD SP is responsible for re-encrypting memory using offline encryption key (OEK), which is generated when a VM is launched. During migration hypervisor initiates a call to MA, which gets all the context using `VM Export API` command which represents internal state of the VM and OEK that is utilized to encrypt VM memory page before migration. AMD SP ensures only relevant content associated with a VM is migrated. MA can only decrypt memory page that are migrated and enforces certain policies related to the migration process. To verify MA associated to a particular

VM, AMD SP remote attestation report also contains measurement of the MA providing more integrity [33].

While AMD SEV SNP proposed a strong security mechanism to be used in cloud environment it suffers from major issues like compatibility, performance overhead and complexity. The encryption and decryption of memory space make the system more secure but at the cost of performance which can concluded from the performance of applications discussed in the next sections. Though AMD SEV protects application from potential vulnerabilities it can't protect if the application in itself is malicious or compromised.

3 Methodology

3.1 Benchmarking

To run a application on a system, we need to be sure of it capabilities it's performance overall which makes benchmarking an integral part of the process. Benchmarking helps developers and architects understand crucial aspects of a system before utilizing them for any high workload application. Benchmarking can be broadly classified as Micro and Macro Benchmarking which are discussed in detail in the next sub section.

3.1.1 Micro Benchmark

Micro benchmarking is defined as a process of measuring a particular function of code or a functionalities by isolating it. Micro benchmarking is employed to measure cloud service performance for various isolated resources such as CPU, I/O, memory, and network [35]. It is usually focused on low level operations which can provide information about performance and also helps in identifying issues related to a system or functionalities. For given research we consider network and CPU benchmarking to check the performance of TEEs using different techniques.

CPU Benchmark is an important benchmark when considering the performance of virtual machine deployed on cloud. As the underneath system configuration changes it affects the performance of any ideal application which needs to be studied to select the best possible environment. SysBench is one such benchmark that helps get a deeper understanding of CPU performance which was used due to its's compatibility with VMs configuration as compared to other CPU benchmarks. It is a cross platform multi threaded benchmarking tool that evaluates performance of a system under intensive load against various OS parameters [36]. Sysbench benchmark suite is utilized in cloud environment to evaluate performance of the system by running various tests on CPU and file I/O across various cloud instance configuration [37]. Sysbench generally runs a defined number of thread and executes all of them simultaneously. Number of request and the type of test mode needs to be defined to execute a particular benchmark on the VM. In case of CPU benchmarking sysbench does a intensive calculations using 64 bit integers that put loads on the system.

One major performance parameter for a system deployed on cloud environments is network performance. As VM continuously interact with other guest VM to build a network for data sharing, network performance becomes a key factor in ensuring proper connections and throughput. Our research here focuses on checking network performance using Netperf [38]. Other Benchmarks like Iperf where considered but due to incompatibility issues and constraints we had to switch to Netperf test. Netperf is a widespread standard benchmarking tool specially designed to measure network performance by considering various parameters like throughput, transmission rate, etc. Netperf supports different communication protocol like TCP and UDP which provide ability to benchmark performance against various configuration. Netperf request response benchmark measures the network performance when two host are communicating. Netperf consist of one client and one server of same configuration interacting with each other for specified time frame where one runs netserver and other is running netperf to benchmark the network. In the given research four different network tests are employed which include TCP stream, UDP stream, TCP RR (Request Response) and UDP RR. TCP stream is used to measure the throughput of a given TCP network whereas UDP stream is used to check throughput of connections under UDP protocol. TCP Request Response test is the measurement of transaction rate between VM under TCP protocol. UDP RR on the other hand considers the same transaction rate under UDP protocol. The mentioned tests help identify network performance of the VMs.

3.1.2 Macro Benchmark

Macro benchmarking is a process of evaluating performance of an application under stress which represents a realistic workloads. It can also be defined as repeatable sequences of tests from end user point of view to test the system under load. Macro benchmarking provides an overview about system performance that is useful for scalability assessment, system load testing and assessing end to end application performance under stress. As macro benchmarking simulates a real world application stress scenario, it equips users with all the necessary information to select the best parameters to help system work under stress. It also helps users identify any potential scalability issues or errors that might show up and restrict the application from performing at optimum capacity in real time. Overall Macro benchmarking provides an overview of different system, application architectures and performance which is crucial when evaluating any sudden software or hardware changes or trying to figure out a particular cloud service provider to deploy your application.

Macro benchmarking was performed in the given research using two application i.e mysql application and high performance computing application. Mysql micro service application was benchmarked using mysqlslap benchmark. Mysqlslap is known as diagnostic application or program specially designed to emulate client load on a mysql server and to report performance of each iteration [39]. Mysqlslap is generally used for testing databases under stress and behaviour of the databases for different level of concurrency and complexity of the queries.

Mysqlslap provides insights on how the databases can handle high traffic. Mysqlslap takes configurable parameters like concurrency and iterations that helps emulate the load on a particular database. Other Benchmarks like HammerDB were also used to check the performance of the Micro Service Applications. Mysqlslap was able to resonate the same benchmarking on all the 4 VMs which made it an ideal choice for this research.

High performance computing was another application that was used to do the macro benchmarking on the TEE VMs [40]. Y-cruncher is the benchmarking tool used to benchmark high performance computing application. Y-cruncher is designed to test performance of computing systems by running high intensive calculations like calculating value of pi to a distinct number of digits which puts intensive loads on any given system or VM [41]. Y-cruncher is preferred in case of HPC applications because calculating computational power and also the efficiency of a system are important parameter when choosing a secure cloud environment. Y-cruncher puts stress on CPU and memory providing a detailed overview of the performance of the VMs. Different benchmarks were tested but running an ideal HPC application required a high end computing VM which was not considered during this research so we had to consider ycruncher with restricted test settings.

3.2 Experiment Setup

The Main aim of the research was to analyze the performance of different trusted execution environments. For this research we considered 4 VMs which included one AMD SEV SNP confidential VM, AMD VM with same configuration as previous one, INTEL TDX confidential VM and one Intel VM [42]. The regions selected for VMs instance were due to limited availability of confidential VM and resources. Below Table 1 provides information about the general configuration and regions where the VMs instances are hosted. In case of AMD NON SEV two VMs were considered one hosted in Japan east and one hosted in India region, due to spot instance error in some parts of implementation.

Virtual Machine	Size	Processor	TEE Enabled	RAM(GB)	Region
INTEL TDX	Standard DC2es-v5	2 core	yes	8	North Europe
INTEL NON TDX	Standard D2s-v3	2 core	no	8	North Europe
AMD SEV	Standard DC2as-v5	2 core	yes	8	Japan East
AMD NON SEV	Standard DC2as-v5	2 core	no	8	Japan East

Table 1. VM Configuration

The VM instances were deployed on Microsoft Azure using the free credits provided under student subscription which also limited scope of project when deciding on different VM instances. Once the confidential VMs were created it was important to validate whether they were able to support TEE.

3.2.1 AMD SEV

AMD SEV Confidential VM [43] was created using Microsoft Azure with default settings and was deployed in japan east region in availability zone set as zone 1. The SSH port number was set to as port no 22 to be accessed using putty.

Once the VM was accessed through ssh it was necessary to verify whether can be used for TEE applications. Initial steps involve enabling SEV mode [44] which involves making changes into the grub file which can be accessed using the command `nano /etc/default/grub.d/50-cloudimg-settings.cfg` and adding `mem_encrypt=on kvm_amd.sev=1` into the grub file as shown in Fig.11.

```
# Cloud Image specific Grub settings for Generic Cloud Images
# CLOUD_IMG: This file was created/modified by the Cloud Image build process

# Set the recordfail timeout
GRUB_RECORDFAIL_TIMEOUT=0

# Do not wait on grub prompt
GRUB_TIMEOUT=0

# Set the default commandlin
GRUB_CMDLINE_LINUX_DEFAULT="modprobe.blacklist=btrfs mem_encrypt=on kvm_amd.sev=1"

# Set the grub console type
GRUB_TERMINAL=console

GRUB_DISABLE_OS_PROBER=false
```

Fig. 11. AMD SEV Grub File

The below command as shown in Fig.12 was used to verify the AMD SEV configuration after changes were implemented into the grub file.

```
root@amdsev:/home/dissertation# cat /proc/cmdline
snapd_recovery_mode=cloudimg-rootfs console=tty1 console=ttyS0 earlyprintk=ttyS0
```

Fig. 12. Checking AMD SEV Configuration

The below Fig.13 shows command which was used to verify AMD SEV SNP status as active.

```
root@amdsev:/home/dissertation# dmesg | grep SEV
[    0.390766] Memory Encryption Features active: AMD SEV
```

Fig. 13. AMD SEV Verification

3.2.2 Intel TDX

Intel TDX VM [45] was created using Microsoft Azure with default settings

and was deployed in the north europe region in availability zone set as zone 1. The SSH port number was set to as port no 22 to be accessed using putty.

Once the VM was accessed through ssh it was necessary to verify whether it can be used for TEE applications. TDX confidential VM comes with TEE mode already in active state which makes the processing easier. The below Fig.14 shows the command used to verify Intel TDX mode.

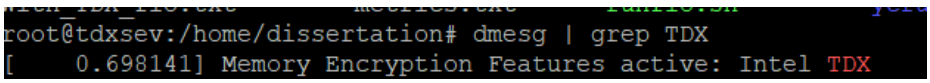

Fig. 14. Intel TDX Verification

The Other 2 VMs were configured with default AMD and Intel setting and deployed in respective regions.

3.3 Implementation

3.3.1 CPU Benchmarking

For this research, we make use of prime number calculation upto a defined limit i.e in current case is 20000 due to limited system capabilities. Calculation of prime number puts intensive load and stresses the CPU, which helps benchmark the performance capabilities of different VMs in cloud. Important feature of sysbench is the use of multi threading, which can be used to check the multi core performance of the system. As we run the test using different number of threads, it gives an insight how performance varies of the system as the CPU scales. For consideration we varied the thread count from 1 to 16 and checked performance across the VMs. The number of total prime number calculations which is completed per second is referred to as CPU's throughput. Sysbench provides a range of parameters that can be altered as per application configuration which includes name of operation which in given research is calculating maximum prime number which can be defined using the given syntax `--cpu-max-prime`, other parameters include duration of test to set a time limit in case of limited CPU resources and thread number which can be varied accordingly. Performance is measured by the time taken for completing the calculations by various machines. Once the benchmarking process is started it continues to run until the given calculation is completed or execution time exceeds the defined test duration.

The below command as shown in Fig.15 shows sysbench benchmark running on 4 threads to calculate prime number till 20000.

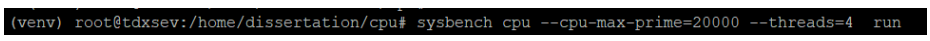

Fig. 15. Sysbench Benchmark

The below Fig.16 shows sysbench benchmark running on 16 threads to calculate prime number till 20000.

```
root@tdxsev:/home/dissertation/cpu# sysbench cpu --cpu-max-prime=2000 --threads=16 run
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 16
Initializing random number generator from current time


Prime numbers limit: 2000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 31802.75

General statistics:
    total time:                          10.0003s
    total number of events:              318070

Latency (ms):
         min:                                    0.03
         avg:                                    0.50
         max:                                   56.09
         95th percentile:                        0.09
         sum:                               159452.67

Threads fairness:
    events (avg/stddev):           19879.3750/536.86
    execution time (avg/stddev):   9.9658/0.02
```

Fig. 16. Sysbench Benchmarking with 4 threads

The above procedure needs to be repeated on each of the VMs (Intel TDX VM, Intel VM, AMD VM, AMD SEV SNP VM) for 5 iterations to get a brief understanding of the average performance of the VMs. We create a bash script to run the benchmarking test multiple times to get a clearer idea of the performance of the VMS. Once the benchmarking script is executed on all 4 VMs, the output is stored as a csv file. The csv files generated on all the 4 VMs are then used by a Python script using the matplotlib library to create a bar plot which can be used in the next section for analysis purposes.

3.3.2 Network Benchmarking

First step to run netperf benchmark is to install all the dependencies associated with the benchmarking that includes - netperf which is responsible for creating a netserver that we can use to establish communication between client and our host application. Network benchmarking here is done using the four tests (Stream, Request Response) as discussed in the previous section. We can also specify the time limit we want the test to take place for which can be varied depending upon the requirement. Benchmarking is done across the Intel and AMD VMs out of which Intel TDX and Intel NON TDX VM are located in the same zone i.e north europe and AMD SEV SNP and AMD VM are located in japan east. It is always preferred that all the VMs in consideration should be located in the similar region, but due to the limited availability of confidential VMs we had to deploy AMD and TDX VMs in 2 different regions, so the comparison in the below benchmarking is divided between Intel TDX and Intel NON TDX VM on

one hand and AMD and AMD SEV SNP VM on the other hand to have a brief understanding of their performance. The region in which the VMs is located also plays a crucial role in affecting the performance of the Network benchmarking.

The below Fig.17 shows the initial step of starting netserver at port 20000 which is used to establish communication between our benchmarking VM and client.

```
(venv) root@tdxsev:/home/dissertation/netperf# netserver -p 20000
Starting netserver with host 'IN(6)ADDR_ANY' port '20000' and family AF_UNSPEC
```

Fig. 17. Starting Netserver

The below command shown in Fig.18 was used to run benchmarking Stream test using TCP protocol on IP 10.0.0.4 communicating on port 20000 for 10 seconds. This test was used to analyze the throughput of the network.

```
(venv) root@tdxsev:/home/dissertation/netperf# netperf -H 10.0.0.4 -p 20000 -t TCP_STREAM -l 10
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 10.0.0.4 () port 0 AF_INET : demo
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

131072  16384  16384    10.00    50494.52
```

Fig. 18. Netperf TCP Stream Test

The below command shown in Fig.19 was used to run benchmarking Stream test using UDP protocol on IP 10.0.0.4 communicating on port 20000 for 50 seconds. This test was used to analyze the throughput of the network.

```
(venv) root@tdxsev:/home/dissertation/netperf# netperf -H 10.0.0.4 -p 20000 -t UDP_STREAM -l 50
MIGRATED UDP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 10.0.0.4 () port 0 AF_INET : demo
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

212992   65507   50.00     6564105      0    68799.08
```

Fig. 19. Netperf UDP Stream Test

The below command shown in Fig.20 was used to run benchmarking RR (Request/Response) test using TCP protocol on IP 10.0.0.4 communicating on port 20000 for 50 seconds. This test was used to analyze the transmission rate.

```
(venv) root@tdxsev:/home/dissertation/netperf# netperf -H 10.0.0.4 -p 20000 -t TCP_RR -l 50
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 10.0.0.4 () port 0 AF_INET : demo : first burst 0
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

16384  131072 1        1       50.00    14467.81
```

Fig. 20. Netperf TCP RR Test

The below command shown in Fig.21 was used to run benchmarking RR (Request/Response) test using UDP protocol on IP 10.0.0.4 communicating on port 20000 for 50 seconds. This test was used to analyze the transmission rate.

```
(venv) root@tdxsev:/home/dissertation/netperf# netperf -H 10.0.0.4 -p 20000 -t UDP_RR -l 240
MIGRATED UDP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 10.0.0.4 () port 0 AF_INET : demo : first burst 0
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

212992 212992 1        1       240.00   26511.62
```

Fig. 21. Netperf UDP RR Test

We create two different bash scripts to benchmark throughput and transmission rate minimum 5 times on different VMs which makes the benchmarking more efficient for reproducibility across the VMs. Once the benchmarking script is executed on all of the 4 VMs, the output is stored as a csv file. The csv files generated on all the 4 VMs are then used by a python script using matplotlib library to create a bar plot which can be used in the next section for analysis purpose.

3.3.3 MySql Microservice Application

Mysql Microservice application is benchmarked using mysqlslap to put load on the database which is used with maximum configuration setting. Running mysql microservice application requires a client and a mysql server which are initially configured along with all the dependencies. Mysqlslap benchmark takes input like concurrency, iterations and queries to put load on our application and provide with benchmarking report. Initial step involves installing dependencies that includes mysqlslap benchmark, mysql server.

Next step involves creating a client and creating a root user for mysql server as shown in Fig.22.

```
(venv) root@tdxsev:/home/dissertation/mysql/mysql_microservice/HammerDB-4.3# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.37-0ubuntu0.24.04.1 (Ubuntu)
```

Fig. 22. Root User Creation

The below Fig.23 shows the steps involved in accessing the server, creating database along with a user to access and providing permission as necessary.

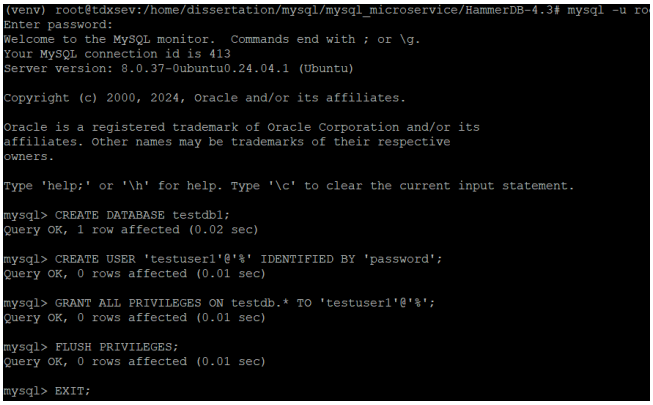

Fig. 23. Creating Database

The below Fig.24 shows the steps involved in creating the table and creating a structure for the table.

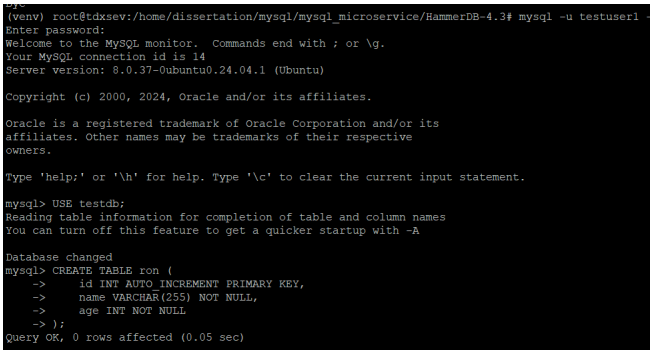

Fig. 24. Creating Table

The below Fig.25 shows steps involved in populating the tables with dummy data to be queried while benchmarking the application.

```
mysql> use testdb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> DELIMITER $$
mysql> CREATE PROCEDURE PopulateUsers1()
    -> BEGIN
    ->     DECLARE i INT DEFAULT 1;
    ->     WHILE i <= 2000 DO
    ->         INSERT INTO ron (name, age) VALUES (CONCAT('ron', i), FLOOR(RAND() * 100));
    ->         SET i = i + 1;
    ->     END WHILE;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql>
mysql> CALL PopulateUsers1();
```

Fig. 25. Populating Database with Dummy Data

The below Fig.26 shows the content of database populated with dummy data.

```
mysql> select * from ron
    -> ;
+------+---------+-----+
| id   | name    | age |
+------+---------+-----+
|    1 | ron1    |  97 |
|    2 | ron2    |  72 |
|    3 | ron3    |  72 |
|    4 | ron4    |  45 |
```

Fig. 26. Querying DB

The below command as shown in Fig.27 is used to benchmark the application by using queries saved in an sql file along with other parameters like concurrency and iterations using mysqlslap.

```
root@dxsev:/home/dissertation/mysql/mysql_microservice/HammerDB-4.3# mysqlslap --user=testuser1 --password=password --host=127.0.0.1 --concurrency=10,20,50,100 --iterations=10 --create-sch
ema=testdb --query=newquery.sql --verbose
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Average number of seconds to run all queries: 0.418 seconds
        Minimum number of seconds to run all queries: 0.356 seconds
        Maximum number of seconds to run all queries: 0.575 seconds
        Number of clients running queries: 10
        Average number of queries per client: 6
```

Fig. 27. Mysqlslap Benchmarking

Once the benchmark runs correctly on all 4 VMs, the cummulative data was being stored and then plotted using matplotlib which was used for comparison and performance analysis.

3.3.4 High Performance Computing Application

High Performance Computing Application benchmarking is done using y-cruncher

benchmark. Y-cruncher benchmark provides a range of parmeters to benchmark any given HPC applications. In the given research y-cruncher was considered to calculate the value of pi till 500m numbers to utilize the maximum capacity of the current VMs. Running other y-cruncher benchmark was not possible on the current VMs due to resource limitation and concurrency issues caused by the VMs.

Initial steps involve downloading and installing y-cruncher benchmark. It involves downloading the official benchmark application from the website and unzipping and installing it in the required directory. After installing a number of files are created which include the executable used to benchmark any given system by running a High performance computing application. The command `./y-cruncher bench all` can be used to enter the benchmarking options to select out of many HPC calculations.

The below Fig.28 shows the execution of y-cruncher for calculation of pi value to digit 500 million.

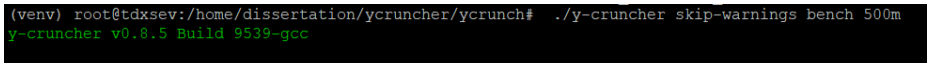

Fig. 28. Ycruncher Benchmark Execution

The below Fig.29 shows the execution of y-cruncher for calculation of pi value to 500 million digit with 4 thread core.

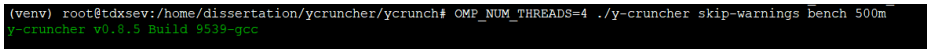

Fig. 29. Y-cruncher Benchmark Running with 4 Threads

The above benchmarking has been automated using a bash script that consider a range of thread number and performs the benchmarking providing with a better overview of the performance. Once the benchmarking script is executed on all of the 4 VMs, the output is stored as a csv file. The csv files generated on all the 4 VMs are then used by a python script using matplotlib library to create a bar plot which can be used in the next section for analysis purpose.

4 Results and Analysis

The below section describes the output of different benchmarking techniques after running on all the 4 VMs. The output is converted into a bar plot to get a comparison between different VMs.

4.1 MYSQL application

The below Fig.30 shows the performance of different TEE VMs when running mysql microservice application.

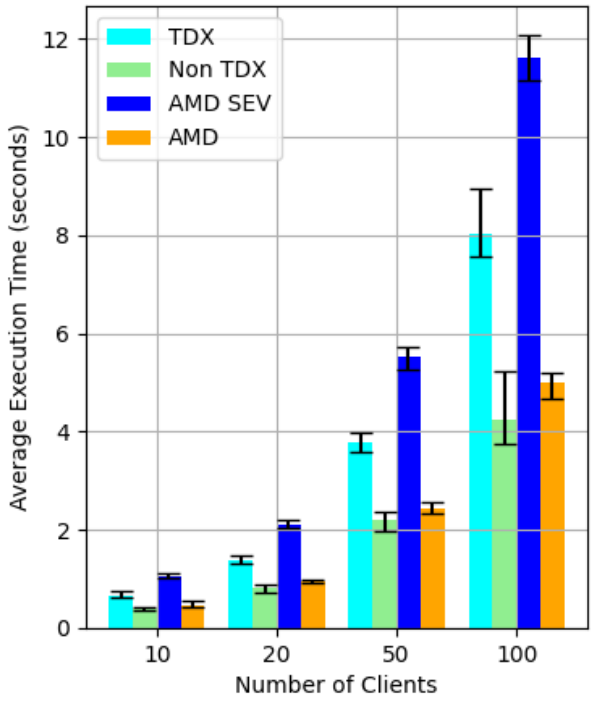

Fig. 30. Mysql Microservice Application Benchmark

As seen from the above plot, it is clear that both TDX and AMD SEV SNP configuration exhibited a relatively higher execution time as compared to the non TDX and AMD machine.The execution time was relatively comparable during initial load which showed the low impact of TEE on applications performance. The execution time increased exponentially incase of confidential VMs whereas non confidential VMs showed a better performance. The increased execution time is due to added security reasons which needs encryption and decryption of application data while processing on confidential VMs. Out of both the confidential VMs Intel TDX showed a better performance, but still relatively poor performance as compared to non confidential VMs. The exponential increase

in the gap between performance of confidential and non confidential VMs also depicts the performance issues at the cost of more secure environment.

4.2 High Performance Computing

Below Fig.31 shows the performance of high performance computing application on 4 different VMs using ycruncher benchmark. Ycruncher benchmark utilizes calculation of value of Pi till 500m. Benchmarking is done by varying thread count from 1 to 16 to get a better performance analysis.

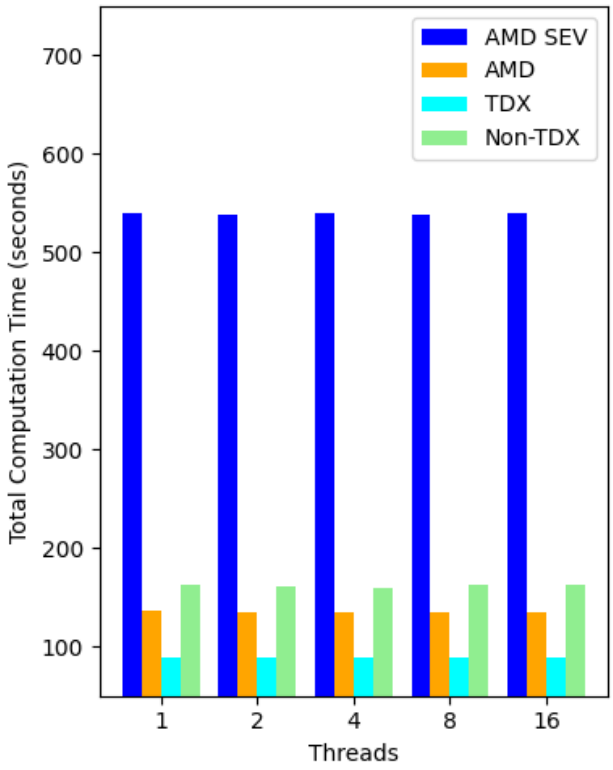

Fig. 31. High Performance Computing Application

AMD SEV SNP exhibited a very high computation time as compared to other VMs when running HPC application which shows the impact of security layer has on the performance of AMD SEV when running a highly computation intensive application. In comparison to AMD TDX and non TDX machine showed a more comparable performance with TDX performing much better even with all the

security aspects proving to be the best performer. AMD VM showed a more linear performance as compared to AMD SEV SNP VM showcasing a lower computation time when not utilizing a secure environment. The performance issue associated with AMD SEV SNP is because of encrypting and decrypting data which leads to high computation timings.

4.3 Network Benchmark

The below Fig.32 shows the netperf throughput benchmark with 2 different test i.e UDP and TCP Stream for all the 4 VMS deployed in separate regions. The plotting and comparison is done between AMD and AMD SEV SNP deployed in japan east region and between Intel TDX and Non TDX deployed in north europe region due to limited availability of confidential VMs.

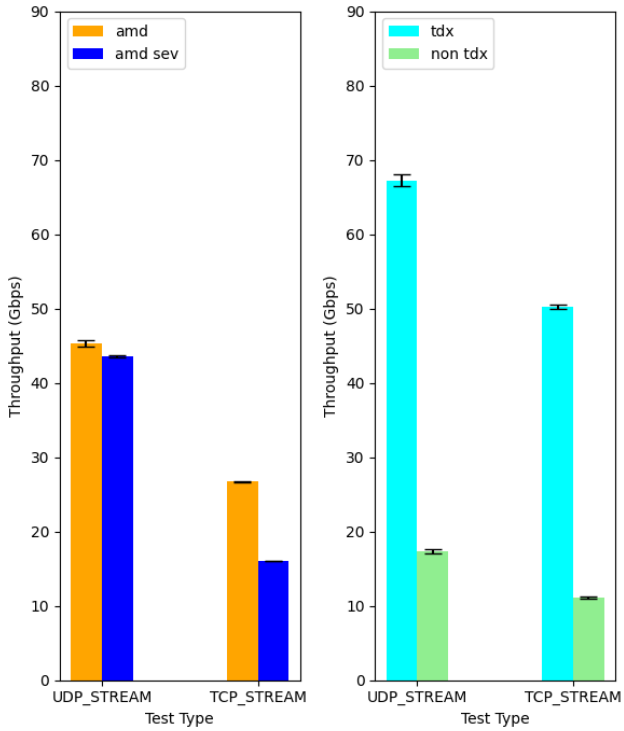

Fig. 32. Netperf Throughput Benchmark

From the throughput results it is clear that between AMD and AMD SEV SNP, AMD VM was performing better than the securely protected AMD SEV SNP because of the added security layer which leads to encryption, decryption of

data that would results in a drop in overall network performance. In case of TDX and non TDX VMs, Intel TDX provided a very high throughput for both TCP stream and UDP stream as compared to non TDX VM. According to the plot TDX VM offers a better optimization for handling network and also reducing latency in processing packets which has resulted in a very high throughput as compared to normal Intel machine.

The below Fig.33 shows the netperf transmission rate benchmark with 2 different test i.e UDP and TCP Request Response test for all the 4 VMS deployed in separate regions. The plotting and comparison is done between AMD and AMD SEV SNP deployed in japan east region and between Intel TDX and Non TDX deployed in north europe region due to limited availability of confidential VMs.

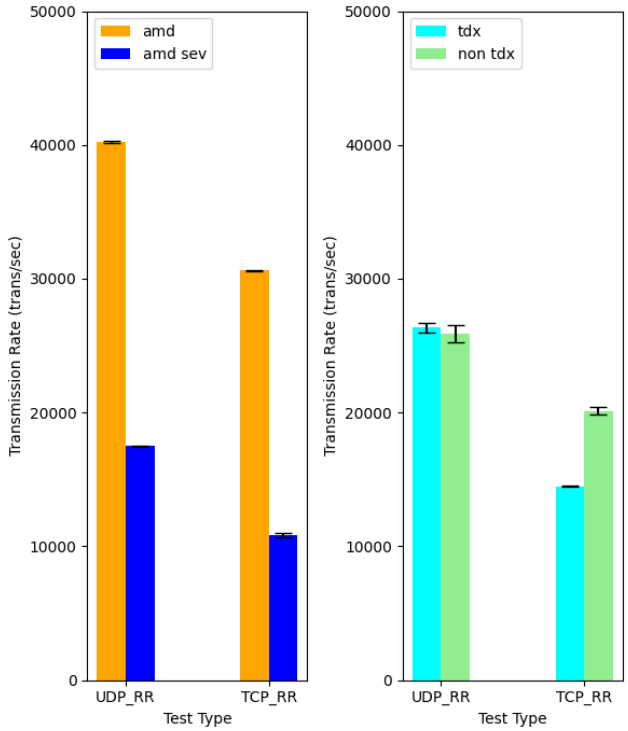

Fig. 33. Netperf Transmission Benchmark

From the above plot it is clear that AMD SEV SNP showed a drop in performance for UDP and TCP Request Response test as compared to AMD VM because of the added security layer in AMD SEV. TDX on the other hand was able to show a similar transmission rate as normal Intel VM which showcases

TDX ability to optimize a secure execution environment without compromising on any network performance.

4.4 CPU Benchmark

The below Fig.34 shows the CPU benchmarking using Sysbench which shows the change in performance of VMs with different thread counts.

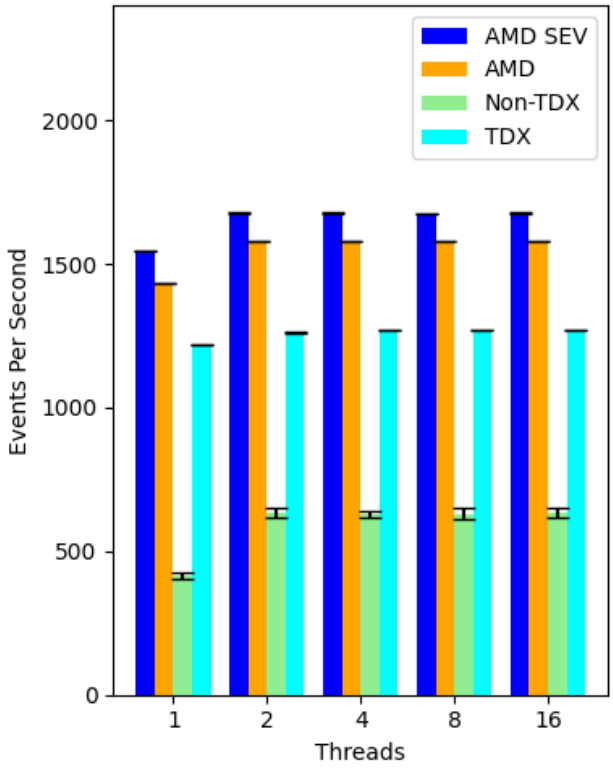

Fig. 34. CPU Benchmarking

From the Fig.34 it is clear that AMD SEV SNP is among the best performers and shows a high events execution per second as thread count increases, which is contrary to the previous benchmarks. Normal AMD machine also shows a good performance which is at par with AMD SEV SNP despite AMD SEV SNP having an added security overhead which makes them suitable for CPU intensive tasks. TDX and NON TDX machine had a huge performance difference which contradicts the previous benchmarks. The CPU performance of TDX enabled

VMs was better than non TDX enabled VM. The whole benchmarking suggest that AMD SEV gets and upper hand to Intel TDX for cpu intensive tasks.

5 Conclusion

As systems are evolving and increasing number of companies are switching to cloud environment. The security challenges associated with the cloud environment are pushing companies to adopt new technologies like trusted execution environment to safeguard their workloads. Even though TEE provides a more secure execution environment, there could be a situation where the performance of the system may be compromised when utilizing the security features. Companies need to conduct a performance analysis of a given application to decide whether a given application needs to be hosted in trusted execution environment for added protection. The given research was focused on finding the performance of different trusted execution environment when trying to execute different microservice applications to understand the trade offs between the security provided to the system and performance of the overall system to help companies make an informed decision while deploying workload in secure environments. Our study involved benchmarking micro service application which included mysql and high performance computing application.

- For mysql application, it was analyzed that confidential VMs performed poorely as compared to non confidential VM and the computing time difference varies exponentially showcasing poor performance of TEE in case of database applications. Which suggested the added security layer incase of confidential VM is hindering the performance of system resulting in higher computation time for encrypting and decrypting data.
- Incase of High performance computing application, we found a change in performance between different TEEs with AMD performing poorly as compared to counterparts. For High performance applications companies need to consider different TEE based on different applications to have a good performance.
- Micro Benchmarking was done to check the CPU and network performance of different TEE, which showed that overall performance of a TEE system varied significantly and selecting an ideal need TEE or non TEE environment depends upon the usage of a particular system application and resource consumption.

From the above pointers it is clear that the performance of TEE varied based on different applications and to select a Trusted Execution Environment for a workload the company needs to consider a wide range of benchmarks and the associated cost to make an informed decision. Our study provides an overview of few applications and benchmarks that will guide a company towards making the selection of a TEE based on companies needs. The Trusted Execution Environment are continuously evolving and as new threats emerge having an TEE environment would be beneficial for a company, to protect the companies resources and having a secure execution environment.

5.1 Future Works

Our research focused on providing companies a guide while selecting a TEE environment for their workload in cloud environment by performing intensive benchmarking which we are planning to present in a conference to publish the research. The current research was restricted by resources and time limitation which can be extended to get a more deeper understanding of the TEE environment as explained below.

- Considering a high quality VMs with similar configurations can be a good way to analyze the performance of the TEE technology enabling more accurate performance analysis. Future research can consider all the VMs of the same size and also located within the same region providing a consistent setup.
- The given research was based on few limited benchmarks but to get an holistic performance analysis wide range of benchmarking tools can be used for parameters like latency, I/O performance, memory benchmarking etc.
- As technology is evolving new applications and integrations are created which create a wider range of applications to be benchmarked, to study the ideal performance of an application based on companies needs.
- Creation of a unified frameworks that can be used to benchmarks different TEE performance and create a standarized evaluation technique that can be used by companies which can be open sourced to help collaborations and improvement in the TEE environments.
- Consideration of other TEE environment like ARM Trustzone and other software implemented TEE which can be studied and benchmarked, which can help companies emulate the environment without any cost overheads.
- Considering different public clouds to get a better understanding of the Confidential VM instances.

The above factors can be considered to extent the scope of the research to keep up with evolving needs of organizations to secure workloads deployed in cloud environment.

Acknowledgments. I would like to thank my dissertation supervisor, Dr Dev Jha, for guiding me through numerous meetings, answering all my questions, helping brainstorm different ideas and keeping project on track by providing constant support. I would also like to thank my family and friends who supported me during the whole dissertation.

Bibliography

[1] Tim Geppert, Stefan Deml, David Sturzenegger, and Nico Ebert. Trusted execution environments: Applications and organizational challenges. *Frontiers in Computer Science*, 4, July 2022.

[2] Markus Sommerhalder. Trusted execution environment. In *Trends in Data Protection and Encryption Technologies*, pages 383–396. Springer, Cham, 2023.

[3] Ieee standard for secure computing based on trusted execution environment. *IEEE Std 2952-2023*, pages 1–29, 2023.

[4] Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi. A comparison study of intel sgx and amd memory encryption technology. New York, NY, USA, 2018. Association for Computing Machinery.

[5] Saima Mehraj and M. Tariq Banday. Establishing a zero trust strategy in cloud computing environment. In *2020 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6, 2020.

[6] Osama Hosam and Fan BinYuan. A comprehensive analysis of trusted execution environments. In *2022 8th International Conference on Information Technology Trends (ITT)*, pages 61–66, 2022.

[7] Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. Trusted execution environments: Properties, applications, and challenges. *IEEE Security Privacy*, 18(2):56–60, 2020.

[8] Luigi Coppolino, Giovanni Mazzeo, and Luigi Romano. Awesome trusted execution environment. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, pages 5–6, 2023.

[9] OMTP. Advanced trusted environment: Omtp tr1 v1.1. Technical report, Open Mobile Terminal Platform, 2009. URL http://www.omtp.org/OMTP_Advanced_Trusted_Environment_OMTP_TR1_v1_1.pdf. Accessed: 2024-06-27.

[10] Xiang Li, Fabing Li, and Mingyu Gao. Flare: A fast, secure, and memory-efficient distributed analytics framework. *Proc. VLDB Endow.*, 16(6): 1439–1452, feb 2023.

[11] Unsung Lee and Chanik Park. Softee: Software-based trusted execution environment for user applications. *IEEE Access*, 8:121874–121888, 2020.

[12] Ayaz Akram, Venkatesh Akella, Sean Peisert, and Jason Lowe-Power. Sok: Limitations of confidential computing via tees for high-performance compute systems. In *2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED)*, pages 121–132, 2022.

[13] Newton C. Will, Tiago Heinrich, Amanda B. Viescinski, and Carlos A. Maziero. Trusted inter-process communication using hardware enclaves. In *2021 IEEE International Systems Conference (SysCon)*, pages 1–7, 2021.

[14] Rui Wang and Yonghang Yan. A survey of secure boot schemes for embedded devices. In *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pages 224–227, 2022.

[15] Ahmad B. Usman, Nigel Cole, Mikael Asplund, Felipe Boeira, and Christian Vestlund. Remote attestation assurance arguments for trusted execution environments. In *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, SaT-CPS '23, page 33–42. Association for Computing Machinery, 2023.

[16] Mengyuan Li, Yuheng Yang, Guoxing Chen, Mengjia Yan, and Yinqian Zhang. Sok: Understanding designs choices and pitfalls of trusted execution environments. 2024.

[17] Luigi Coppolino, Salvatore D'Antonio, Valerio Formicola, Giovanni Mazzeo, and Luigi Romano. Vise: Combining intel sgx and homomorphic encryption for cloud industrial control systems. *IEEE Transactions on Computers*, 70 (5):711–724, 2021.

[18] Somnath Chakrabarti, Thomas Knauth, Dmitrii Kuvaiskii, Michael Steiner, and Mona Vij. Chapter 8 - trusted execution environment with intel sgx. In Xiaoqian Jiang and Haixu Tang, editors, *Responsible Genomic Data Sharing*, pages 161–190. Academic Press, 2020.

[19] Tobias Cloosters, Michael Rodler, and Lucas Davi. TeeRex: Discovery and exploitation of memory corruption vulnerabilities in SGX enclaves. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 841–858, August 2020.

[20] Intel. *Remote Attestation of Intel SGX*, 2022. URL https://sys.cs.fau.de/extern/lehre/ws22/akss/material/intel-sgx.pdf. Accessed: 2024-07-10.

[21] Newton C. Will and Carlos A. Maziero. Intel software guard extensions applications: A survey. 55(14s), jul 2023.

[22] Carlton Shepherd and Konstantinos Markantonakis. *Enclave Computing*, pages 133–163. Springer International Publishing, Cham, 2024.

[23] Hasini Witharana, Debapriya Chatterjee, and Prabhat Mishra. Verifying memory confidentiality and integrity of intel tdx trusted execution environments. In *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 44–54, 2024.

[24] Roberto Guanciale, Nicolae Paladi, and Arash Vahidi. Sok: Confidential quartet - comparison of platforms for virtualization-based confidential computing. In *2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED)*, pages 109–120, 2022.

[25] Alexander Eichhorn. *AMD SEV and Intel TDX*, 2022. URL https://sys.cs.fau.de/extern/lehre/ws22/akss/material/amd-sev-intel-tdx.pdf. Accessed: 2024-07-13.

[26] Intel. *Architecture Specification: Intel Trust Domain Extensions (Intel TDX) Module*, 2022. URL https://cdrdv2.intel.com/v1/dl/getContent/733568. Accessed: 2024-07-14.

[27] Pranav Mishra et al. An in-depth look into intel tdx. *arXiv preprint arXiv:2303.15540*, 2023. URL https://arxiv.org/pdf/2303.15540. Accessed: 2024-07-13.

[28] Luca Wilke, Jan Wichelmann, Anja Rabich, and Thomas Eisenbarth. Sev-step: A single-stepping framework for amd-sev. *arXiv preprint arXiv:2307.14757*, 2023.

[29] Mingjie Yan and Kartik Gopalan. Performance overheads of confidential virtual machines. In *2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–8, 2023.

[30] AMD. Amd memory encryption: Sme and sev, 2018. URL https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/memory-encryption-white-paper.pdf. Accessed: 2024-07-16.

[31] Luca Wilke, Jan Wichelmann, Mathias Morbitzer, and Thomas Eisenbarth. Sevurity: No security without integrity : Breaking integrity-free memory encryption with minimal assumptions. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1483–1496, 2020.

[32] AMD. Protecting vm register state with sev-es. Technical report, Advanced Micro Devices, Inc., year. URL https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/Protecting-VM-Register-State-with-SEV-ES.pdf. Accessed: 2024-07-18.

[33] Robert Buhren. *Resource control attacks against encrypted virtual machines*. PhD thesis, Dissertation, Berlin, Technische Universität Berlin, 2022, 2022.

[34] Pedro Antonino, Ante Derek, and Wojciech Aleksander Wołoszyn. Flexible remote attestation of pre-snp sev vms using sgx enclaves. *IEEE Access*, 11: 90839–90856, 2023.

[35] Joel Scheuner and Philipp Leitner. Estimating cloud application performance based on micro-benchmark profiling. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 90–97, 2018.

[36] Alexey Kopytov. Sysbench: Scriptable database and system performance benchmark, 2024. URL https://github.com/akopytov/sysbench. Accessed: 2024-07-22.

[37] Sanjay P Ahuja and Niharika Deval. System level benchmarks for the cloud. *Computer and Information Science*, 8(2):58, 2015.

[38] The Netperf Homepage. The netperf homepage. URL http://www.netperf.org. Accessed: 2024-07-22.

[39] Oracle Corporation. mysqlslap — a diagnostic program, 2024. URL https://dev.mysql.com/doc/refman/8.0/en/mysqlslap.html. Accessed: 2024-07-22.

[40] Keke Chen. Confidential high-performance computing in the public cloud. *IEEE Internet Computing*, 27(1):24–32, 2023.

[41] Alexander J. Yee. y-cruncher - a multi-threaded pi-program, 2024. URL http://www.numberworld.org/y-cruncher/. Accessed: 2024-07-22.

[42] Microsoft Corporation. Azure confidential computing deployment models, 2024. URL https://learn.microsoft.com/en-us/azure/confidential-computing/

confidential-computing-deployment-models. Accessed: 2024-07-29.

[43] AMD. Amd epyc processors on microsoft azure, 2024. URL https://www.amd.com/en/products/processors/server/epyc/microsoft-azure.html. Accessed: 2024-07-29.

[44] OVHcloud. Dedicated servers with amd sev, 2024. URL https://help.ovhcloud.com/csm/en-gb-dedicated-servers-amd-sme-sev?id=kb_article_view&sysparm_article=KB0044014. Accessed: 2024-07-29.

[45] Intel Corporation. Intel trust domain extensions, 2024. URL https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html. Accessed: 2024-07-29.