# Performance Analysis and Optimization of Confidential Virtual Machines

Luca Mathias
Advisor: Dr. Masanori Misono
Chair of Computer Systems
https://dse.in.tum.de/

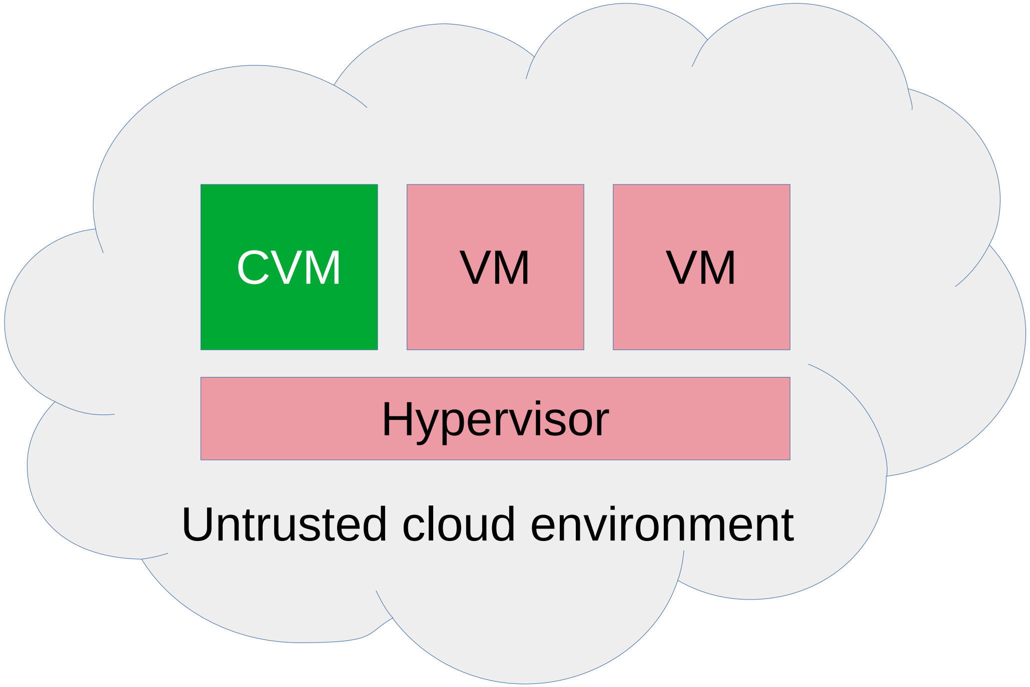
17.06.2024 – 31.10.2024

# Motivation

- Confidential VMs (CVMs) allow for **safe computation on sensitive data**
- Workloads do **not** have to be modified
- Supported on many CPU architectures
  - **AMD SEV-SNP(x86)**, **Intel TDX(x86)**, ARM CCA, PowerPC …

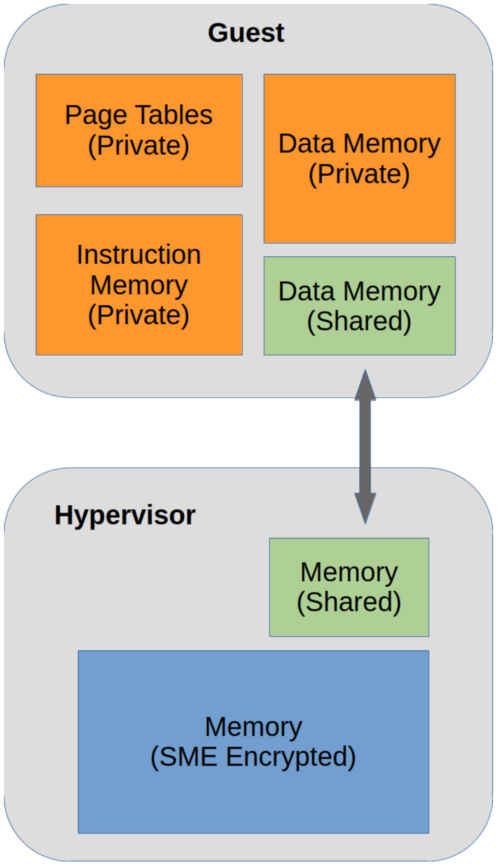
**Our goal:**

- Analyze CVM overheads across different workloads **(network I/O and computational)**
- Examine a simple optimization technique in the form of **CPU polling**



# CVM Overhead Sources

- Main memory and CPU state are **encrypted and protected from unauthorized host/hypervisor access**.
- New **VMEXIT** procedures to protect CPU state.
- **VMGEXITs** and **TDCALLs** cause **severely increased VMEXIT cost**.
- **Bounce buffer mechanism** (swiotlb) enables I/O at the cost of additional data copies



# State-of-the-art

- “**Bifrost**: Analysis and Optimization of Network I/O Tax in Confidential Virtual Machines”
  - Limited to AMD SEV-ES and simulated Intel TDX
  - No examination of system metrics
- “**Confidential VMs Explained**: A Cross-Layer Analysis of AMD SEV-SNP and Intel TDX”
  - Introduces CPU polling as an optimization strategy
  - No examination of system metrics

# Research Gap

- Scope mostly limited to **discussing the results** without deeper analysis
- **No system/hardware metrics** analyzed
- **No remote networking** for networking benchmarks

# Examined Metrics

- **Hardware counters** (SNP only)**:**
  - Instruction count, branch misses, TLB misses, L1 misses
- **VMEXIT count** (TDX and SNP) **and reasons** (SNP only)**:**
  - E.g., MMIO, MSR, VMMCALL, HLT
- **MMIO and MSR addresses**

# Benchmarks

- **Networking benchmarks**
  - Latency (ping)
  - TCP and UDP throughput (iperf)
  - In-memory database server performance (redis/memchached and memtier-benchmark)
  - Web server performance (nginx and wrk)
- **Computational benchmarks**
  - Highly parallel computation (Nas Parallel Benchmark ua)
  - Machine learning (TensorFlow BERT)

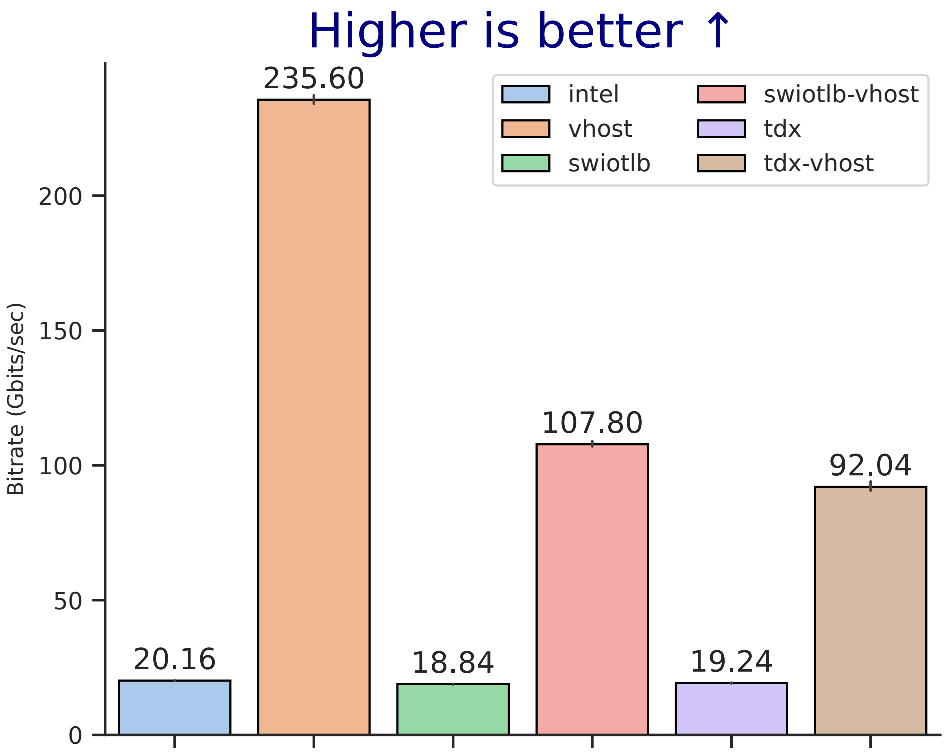
# Analysis

- We analyze **multiple categories of VMs**
  - E.g., with and without vhost and bounce buffer
- **CVMs are compared against traditional VMs** as a baseline
- We examine the effects of **idle and halt CPU polling**

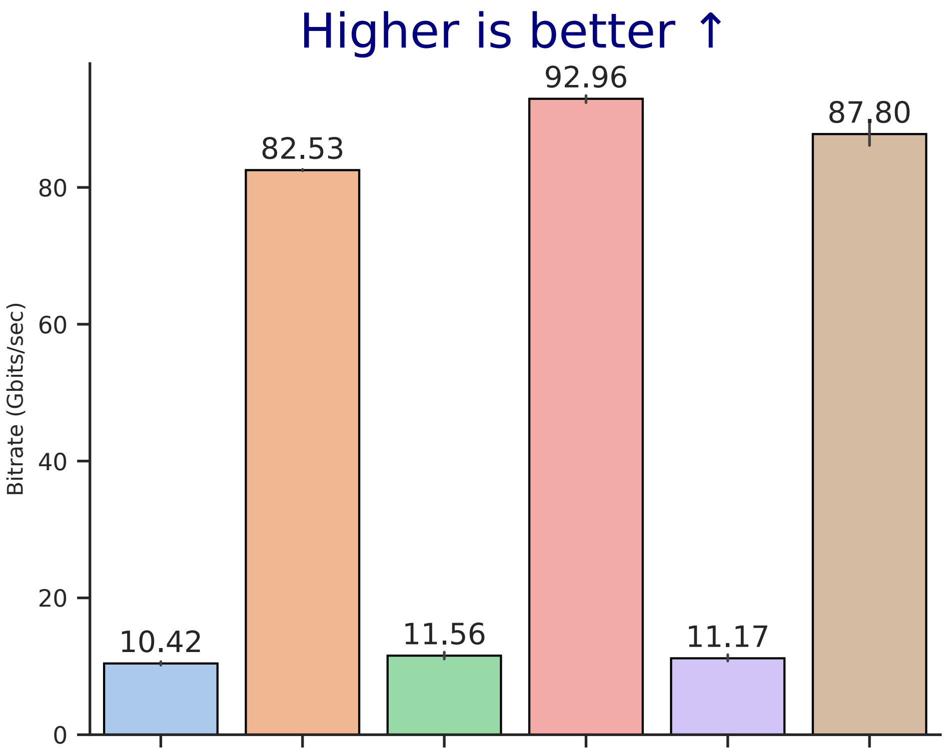
| Identifier | Description |
|---|---|
| intel/amd | Baseline VM |
| snp/tdx | Confidential VM |
| vhost | Vhost protocol enabled |
| swiotlb | Standard VM with BB |
| poll | Idle polling enabled |
| hpoll | Halt polling enabled |

Different VM categories

# Bounce Buffer



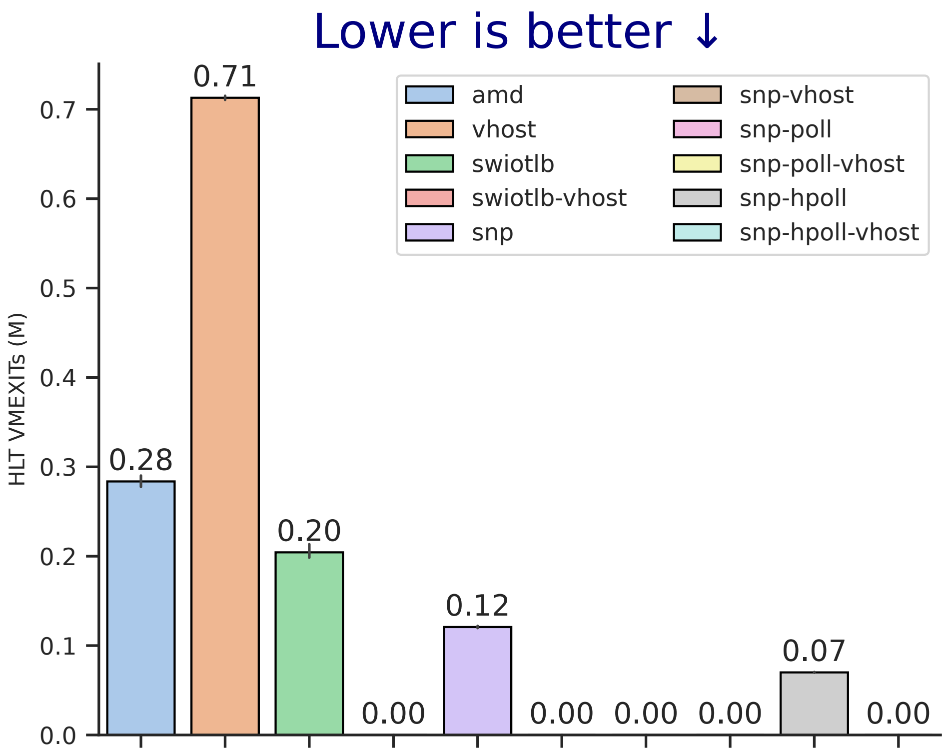

**Bounce buffer bottlenecks on high CPU load** (TCP throughput, local, TDX)




**Bounce buffer can improve performance** (TCP throughput, remote, TDX)

# HLT VMEXITs

- Additional load generated by CVMs and the bounce buffer can **decrease the amount of HLT-related VMEXITs**
- **Similar effect as CPU polling**
- Can lead to **increased performance for CVMs under low CPU load**

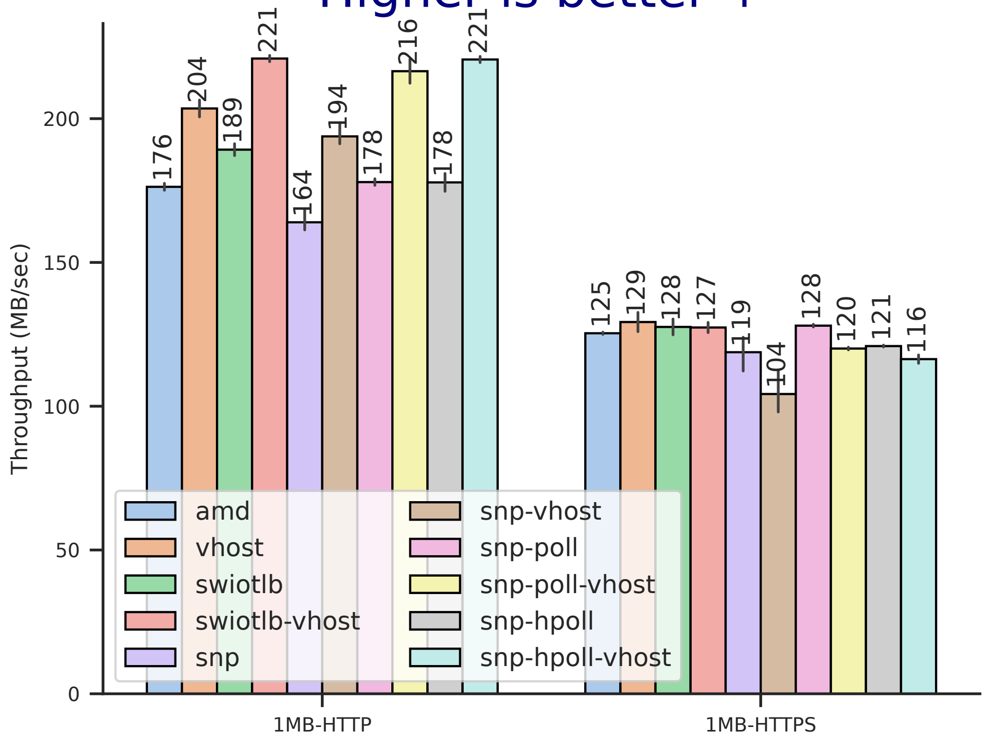



**SEV-SNP, local, TCP throughput**

# General CVM Overheads

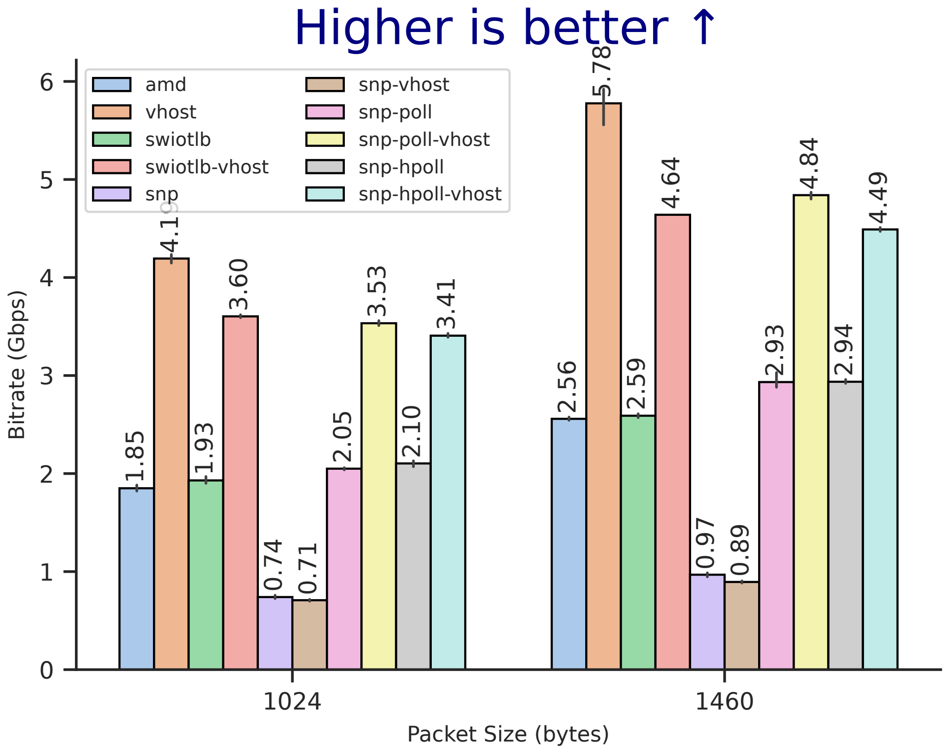
- SNP and TDX normally show significant but non-excessive **overhead of between 5% and 20% in most cases**
- These are **not caused by the bounce buffer mechanism**
- They are **caused by VMEXIT behavior** and **significantly reduced by CPU polling**
- In some scenarios **extreme overheads of over 80%** are encountered

Higher is better ↑

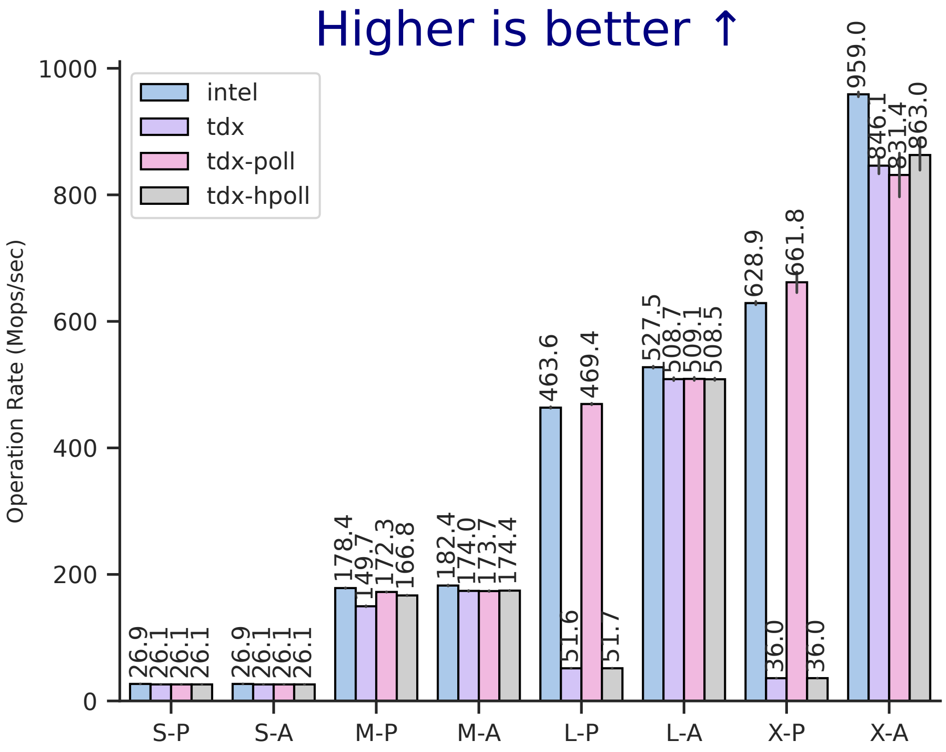



**SEV-SNP web server performance**

# Extreme Overheads




**Overheads of over 80% for UDP throughput** (SEV-SNP, local)



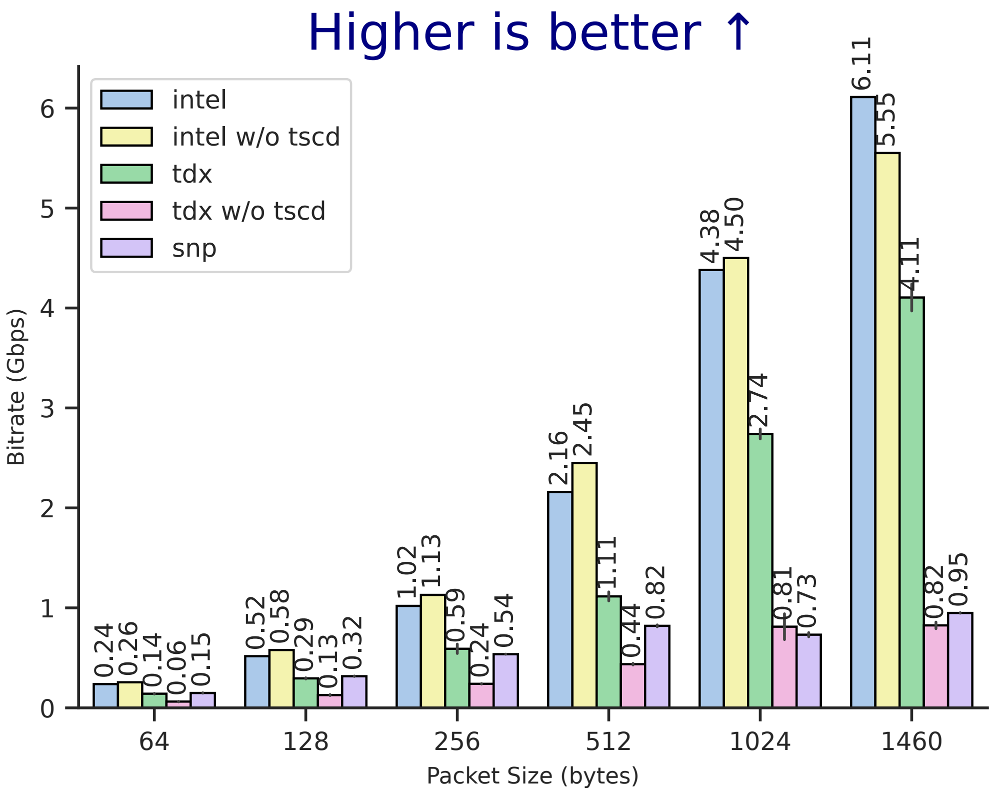

**Massive overheads for highly parallel computing** (TDX)

# TSC-Deadline Mode

- SNP does not support **TSC-Deadline APIC operation mode** (**MSR Interface**)
- Legacy APIC programming modes are used (**MMIO Interface**)
- They offer **worse time resolution (local clock instead of CPU clock)**, which seems to lead to **more frequent reprogramming**
- This causes a **high amount of MMIO VMEXITs instead of fewer MSR VMEXITs**
- This **severely decreases performance** in scenarios with a non-linear execution flow or many synchronization events

# Example

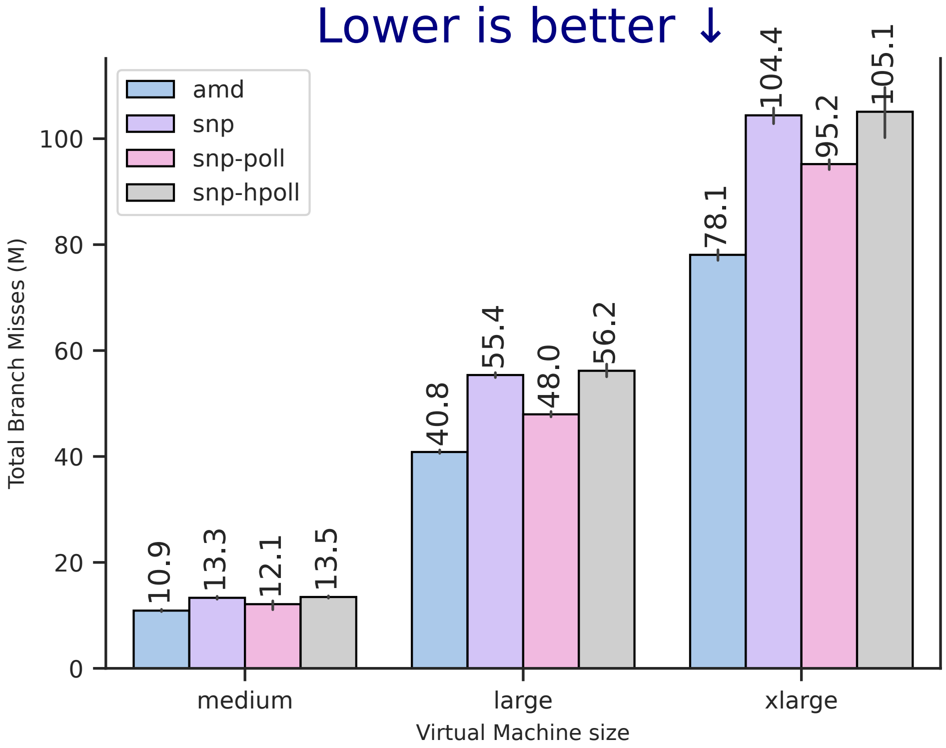
- TDX **does** support TSC-Deadline mode
- **Performance drops significantly if it is disabled**
- **VMEXITs increase by nearly 2x**
- Traditional Intel VMs are **less effected**




**UDP bitrate with and without TSC-Deadline mode.**

# Hardware Events

- During most benchmarks there are **no abnormalities**
- The machine learning benchmark features **increased branch misses**
- AMD SEV-SNP **performs more BTB flushes** in some cases
- L1 cache and TLB misses are not relevantly increased in any scenario




**Machine learning (TensorFlow SEV-SNP)**

# Summary

- **Significant overheads across network I/O and computational performance**
- In most scenarios, **overheads can be attributed to VMEXIT behavior**
- There are **extreme overhead scenarios (>80%)** that seem to be related to local APIC programming
- **CPU polling massively increases CVM performance** in these scenarios
- Under high CPU utilization the bounce buffer **is a performance bottleneck**
- Under low CPU utilization the bounce buffer **can increase performance**
- **Hardware events**, such as L1 misses, play a secondary role and are **not a big factor in CVM overheads** (branch misses are increased in some cases)

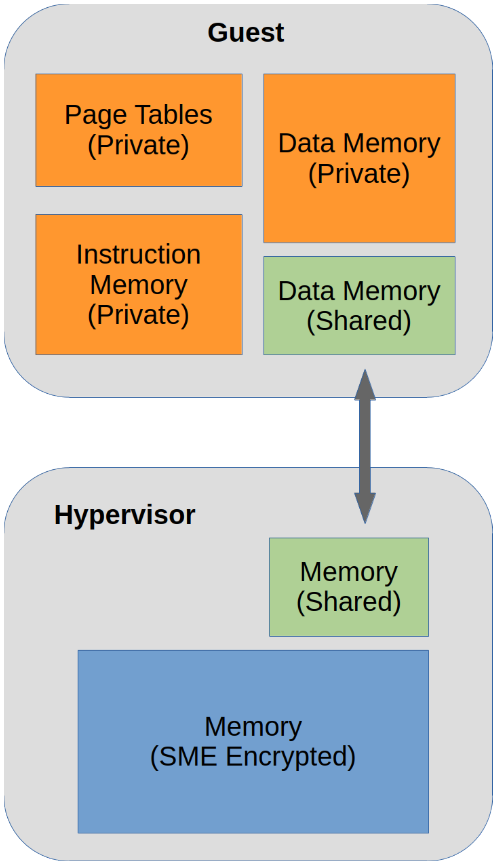
# Future Work

- Resolution of **TDCALLs**
- **Hardware events** for TDX
- **Vhost-user** and different **halt polling parameters**
- **Most promising:** Deeper analysis of high overhead scenarios using **micro-benchmarks**

# Backup

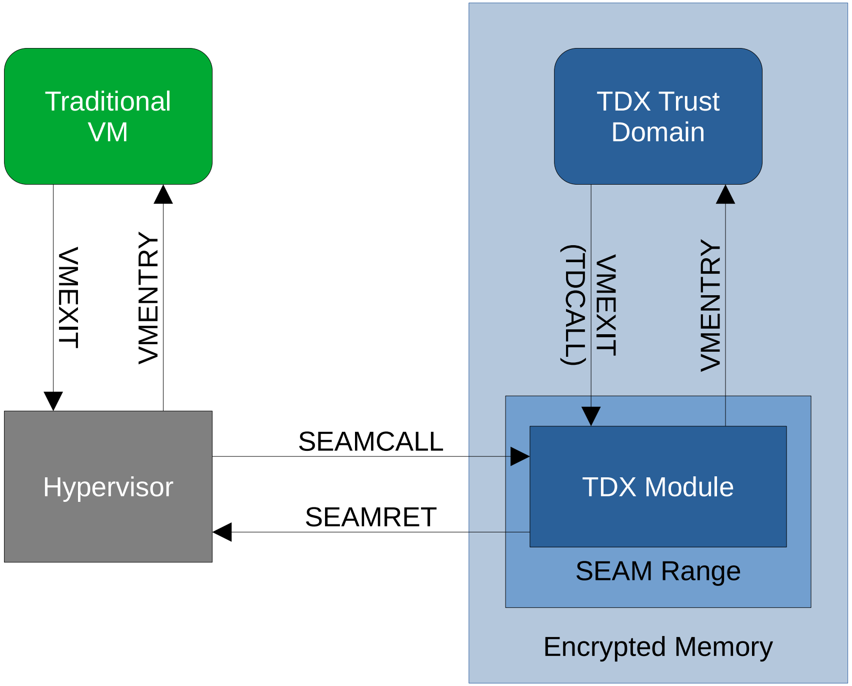
# AMD SEV-SNP VMEXIT

- **Automatic and Non-automatic exits** (most are **NAEs**)
- NAEs add **additional handler overhead** and perform an AE using the **VMGEXIT** instruction
- CPU state transfer/protection is handled by newly introduced data structures (**GHCB** – Guest Hypervisor Control Block)



# Intel TDX VMEXIT

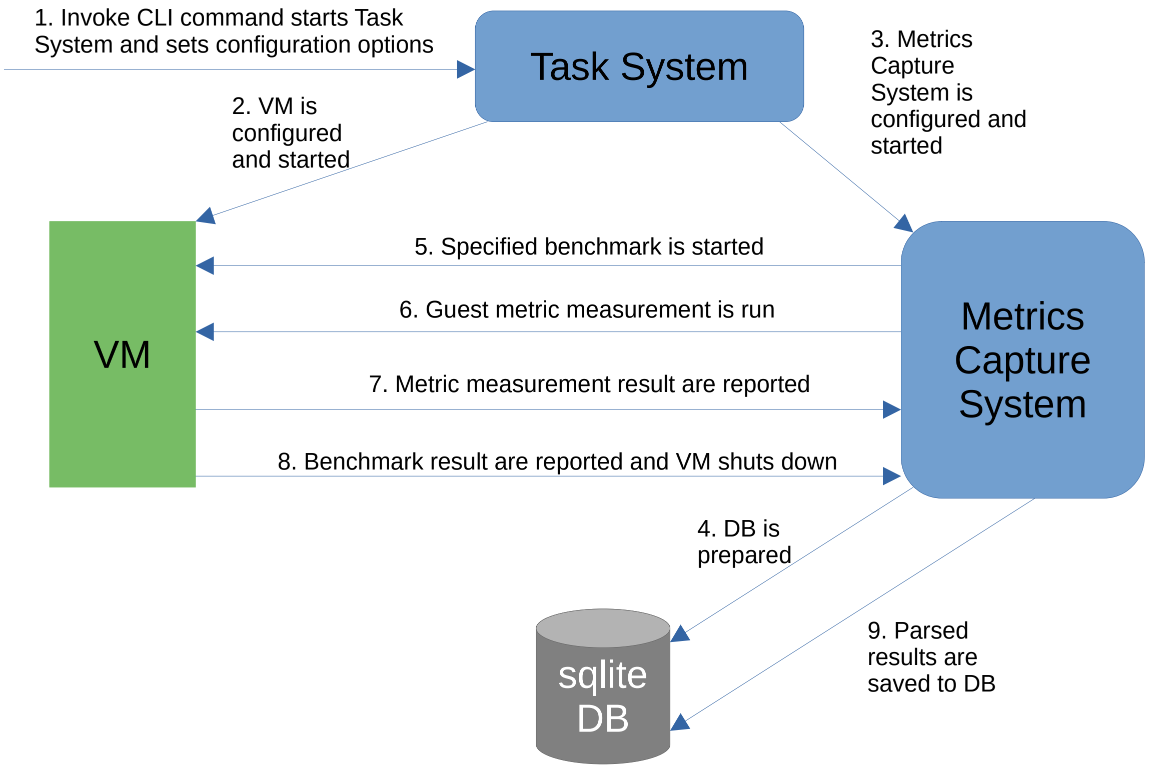
- Intel **TDX module acts as intermediary** between the TD/CVM and the hypervisor
- **VMEXITs** are performed via a **TDCALL** instruction with additional overhead
- CPU State transfer/protection is managed via dedicated memory pages by the TDX module



# Data Capture System

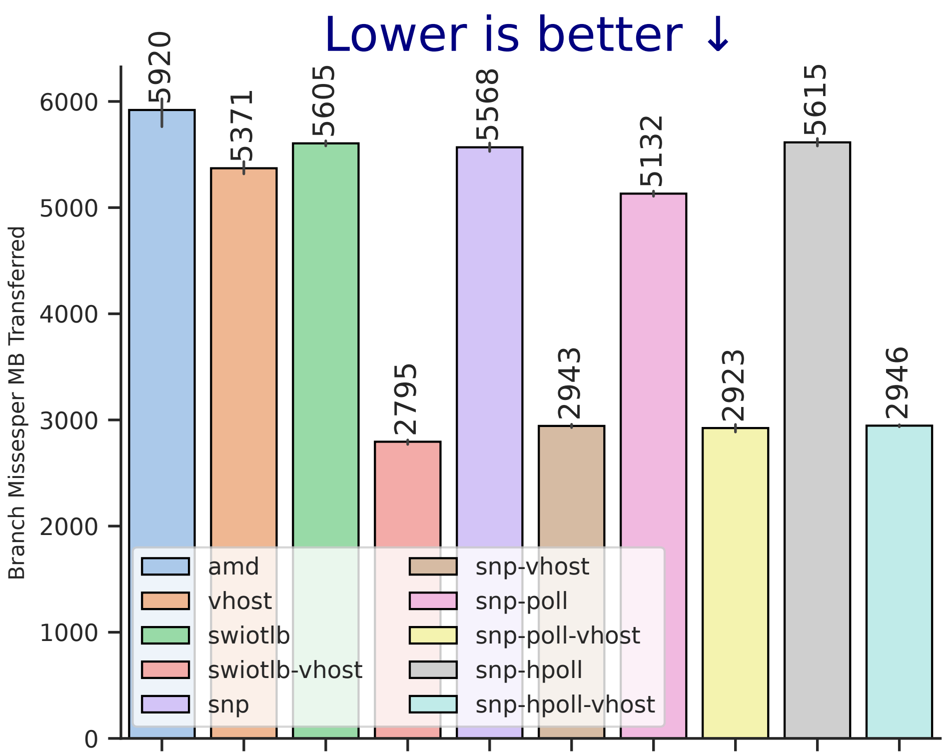
We use **PyInvoke** to **automate:**

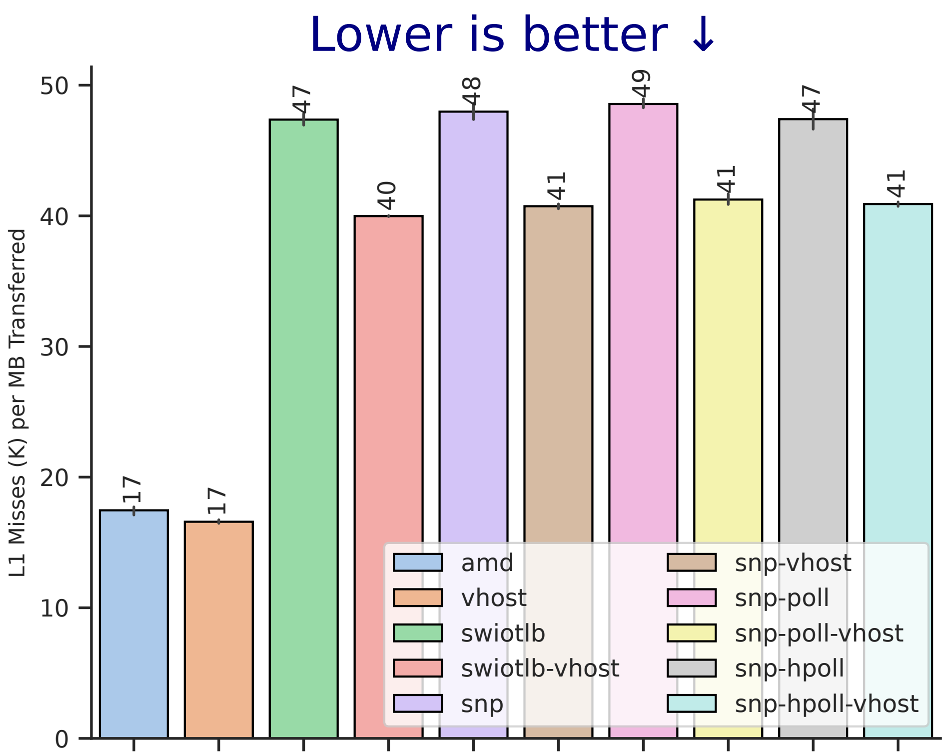
- Benchmark execution
- Benchmark data capture
- System metrics capture
- Data organization




PyInvoke: https://www.pyinvoke.org/

# Additional Hardware Counter Examples




**Branch misses and L1 cache misses during TCP throughput benchmark**