Towards GPU Passthrough in Intel TDX: Design Challenges and Early Baselines

Yoshiaki Sato
Waseda University
yoshisato@kasahara.cs.waseda.ac.jp

Hidetoshi Uranami
Waseda University
reinsirk@kasahara.cs.waseda.ac.jp

Akihiro Saiki
Waseda University
saiki@kasahara.cs.waseda.ac.jp

Keiji Kimura
Waseda University
keiji@waseda.jp

***Abstract*—Confidential Virtual Machines (CVMs) like Intel TDX and AMD SEV-SNP provide hardware-based isolation for cloud workloads. While SEV-SNP supports GPU passthrough for legacy (non-CC capable) accelerators, Intel TDX currently lacks this capability. This work-in-progress analyzes the performance implications of enabling such functionality across both platforms. We identify fundamental architectural differences: SEV-SNP's guest-controlled page encryption enables near-native performance, while TDX's hypervisor-mediated design requires mandatory VMEXITs and may impose significantly higher costs. Our CUDA microbenchmarks establish baseline metrics showing pageable transfers achieve 50-75% throughput of pinned DMA on Intel Xeon systems. We present the technical requirements for TDX GPU passthrough and analyze why Intel's stricter security model complicates performance. This work provides developers with critical performance expectations when choosing between CVM platforms for GPU workloads, highlighting the trade-offs between security architectures and practical performance.**

***Index Terms*—Confidential Computing, Intel TDX, NVIDIA GPU**

I. Introduction and Motivation

The rapid ascent of AI and large–scale data analytics is tilting cloud usage ever further toward GPU–accelerated workloads. Confidential computing (CC) offerings such as Intel TDX and AMD SEV-SNP extend hardware isolation to these multi-tenant clouds, yet GPU support inside a confidential VM (CVM) remains emerging. While privacy-preserving LLM inference and federated learning demonstrably benefit from TEEs [1]–[4], today only NVIDIA H100 ships with on-board mechanisms for encrypted DMA and device attestation [5], [6]. This leaves the vast installed base of pre-Hopper cards unusable inside CVMs. While many projects evaluate the performance of H100 GPUs in CC-scenarios, to the best of our knowledge, no projects have evaluated the performance of non-CC GPUs inside CVMs [7]–[9].

Recent proof-of-concept patches show that non-CC GPUs can operate in a SEV-SNP guest by selectively decrypting memory regions used for GPU DMA access while maintaining encryption for the rest of the VM's memory through kernel-level page table modifications and coordination with the AMD Secure Processor [10]. Intel TDX lacks this stop-gap. This paper closes the first part of the gap by quantifying the initial overheads using micro-benchmarks and discussing the mechanisms and fundamental micro-architectural differences present between SEV-SNP and TDX, informing the implications of non-CC GPU passthrough into TDX. Such a baseline and analysis will show whether TDX's complex memory acceptance protocol imposes performance penalties for GPU support. Our contributions are summarized as follows:

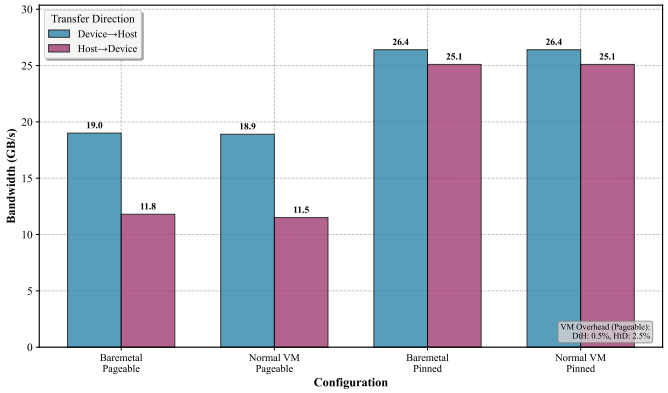

Fig. 1. GPU Data-Transfer Performance on TDX-Capable Intel Xeon Hosts

- Baseline measurement for standard Intel XEON Systems.
- Analysis of the GPU passthrough method in SNP.
- Architectural differences for TDX and its implications.

II. Technical Approach

A. Baseline Measurements on Intel XEON Systems

To establish a baseline for GPU performance, we measured host-to-device and device-to-host copy bandwidth on an NVIDIA GPU (RTX A2000) using cuda-samples bandwidthTest [11]. Two scenarios are tested in Figure 1: a bare-metal Linux host and a normal QEMU VM with GPU passthrough (no CVM features enabled). As shown above, bare-metal and normal VM both achieved an average of 25.8 GB/s bi-directional data transfer rate for pinned memory. On the other hand, pageable transfer rates were noticeably less for both baremetal and normal VM, caused by an extra copy of the memory in a software bounce buffer. The key observation is that even without any memory encryption, pageable transfers impose a 25-50% bandwidth penalty. While a 25-50% bandwidth penalty may appear severe, many ML workloads are compute-bound rather than I/O-bound, suggesting that GPU acceleration remains viable for a range of confidential workloads. Establishing these upper and lower bounds for bandwidth performance creates a foundation against which

upcoming TDX numbers can be judged proving a baseline missing from current literature.

B. Non-CC GPU Passthrough for SEV-SNP

To enable non-CC capable GPUs in AMD SEV-SNP confidential VMs, Hur et al. [12] implemented a two-part patching approach. The kernel patch extends the x86 memory management subsystem with three functions: `is_vm_encrypted()`, `set_vm_decrypted()`, and `set_vm_encrypted()`, which manipulate the encryption bit in page table entries (PTE) and coordinate with the AMD Platform Security Processor (PSP). The NVIDIA open GPU kernel module (OGKM) patch integrates these APIs into the Unified Virtual Memory (UVM) subsystem, automatically decrypting memory regions when external mappings are created and re-encrypting them upon cleanup. This approach creates a controlled security boundary where only GPU-accessible memory regions are selectively decrypted, maintaining the confidentiality of the remaining VM memory.

C. Analysis

This "page-flip method" for SNP inherently trades security for performance as the whole cycle requires zero hypervisor involvement and only micro-second setup latency:

1) **Clear the C-Bit in PTE**: The guest marks the page "shared" by writing *C = 0* in its own page-table entry, telling the memory controller the data no longer needs to be encrypted. [12]
2) **Validate with `PVALIDATE`**: The guest immediately executes the new `PVALIDATE` instruction on that page, which flips the RMP entry from *Guest-Invalid* to *Guest-Valid*. No hypervisor call or VMEXIT is involved. [12]
3) **Flush local TLB**: A quick TLB flush guarantees every core sees the updated C-bit costing only a few µs system-wide.
4) **DMA at wire-speed**: Once the page is *Guest-Valid & C = 0*, the memory controller simply bypasses AES-XTS, so the device and the CPU both see raw DRAM and transfers run at normal PCIe bandwidth. [12]
5) **Flip back when done**: To reclaim the page, the guest sets C = 1 in the PTE and issues `PVALIDATE` again to restore *Guest-Valid & Encrypted* status, and the page is private once again. [10]

This architectural efficiency—zero VMEXITs, guest-controlled validation, and direct memory controller bypass—translates to substantial performance benefits. Recent measurements by Uranami et al. demonstrate that SEV-SNP achieves approximately 22.0GB/s data transfer throughput for GPU workloads [13], approaching the 26.9GB/s baseline measured for regular baremetal transfers. This modest (18%) overhead demonstrates that the page-flip method preserves near-native GPU performance.

D. Non-CC GPU Passthrough for TDX

There are several micro-architectural hurdles when trying to reuse the SNP "page-flip" method on Intel TDX. The fundamental obstacle is caused by the Intel TDX's design that changes architectural control ownership. While SNP gives the guest a one-instruction privilege to validate its own RMP entries, TDX insists the hypervisor mediate every page-state change through Secure-EPT and the TDX module [14], [15]. This adds extra exits, bookkeeping, and a whole *accept* round-trip:

1) **Encryption bit ownership**: For SNP, the guest's own PTE has the C-bit, and clearing it makes the memory controller skip AES-XTS [12]. For TDX, the guest has only an S-bit hint while the real encryption state lives in a Secure-EPT entry that the *TDX module*–not the guest–owns [16]. Therefore, TDX guest cannot simply "flip a bit."
2) **Page-state transition path** (Validation vs. Map/Accept): For SNP, the guest flips C-bit in its PTE and finishes with a *guest-only* `PVALIDATE` instruction, and thus, no exit to the hypervisor is required. However, for TDX to convert page mappings, the TD guest must issue `TDCALL <TDG.VP.VMCALL MapGPA>` for the hypervisor to add a shared mapping and `TDG.MEM.PAGE.ACCEPT` for the TDX module to update the owner bits. Essentially, TD calls MapGPA → TDX-module mediates → hypervisor updates page tables → control returns to TD, adding 2 whole VMEXITs to the workflow [16].

For the proposed passthrough plan, Linux already exposes kernel API that would make implementation effort fairly simple by using `set_memory_decrypted()` and `set_memory_encrypted()` [17]. Therefore, a smart refactor of the Linux kernel patch would be sufficient. However, internally, these helpers wrap the MapGPA + Accept flows, may sleep/retry, and therefore cost far more than the SNP path.

III. Discussion & Next Steps

Our analysis reveals fundamental architectural differences in how TDX and SEV-SNP approach memory security that directly impact GPU passthrough performance. While SEV-SNP's guest-controlled C-bit enables microsecond-scale page transitions, TDX's hypervisor-mediated model introduces mandatory VMEXITs and TDX module calls that may significantly increase overhead. The implications extend beyond performance. TDX's design philosophy—requiring hypervisor involvement for all page state changes—reflects Intel's stricter security model, where the guest cannot unilaterally declassify memory.

Moving forward, implementing our proposed design will quantify these overheads precisely. Key measurements will include: (1) MapGPA latency compared to SEV's PVALIDATE, (2) impact of TDX's accept protocol on streaming DMA workloads, and (3) whether batching page conversions can amortize the VMEXIT costs. These metrics will inform whether TDX's security-first architecture imposes acceptable overhead for GPU-accelerated confidential computing. While both platforms ultimately expose plaintext to achieve

GPU functionality, understanding their performance characteristics helps developers choose appropriate platforms for their security-performance requirements until hardware-based confidential GPU solutions mature.

ACKNOWLEDGEMENT

A part of this paper is supported by JSPS KAKENHI Grant Number JP23K11040.

REFERENCES

[1] K. G. Narra, Z. Lin, Y. Wang, K. Balasubramaniam, and M. Annavaram, "Privacy-Preserving Inference in Machine Learning Services Using Trusted Execution Environments," *arXiv preprint* arXiv:1912.03485, Dec. 2019.

[2] T. Lee *et al.*, "Occlumency: Privacy-Preserving Remote Deep-learning Inference Using SGX," in *Proc. 25th ACM Int. Conf. Mobile Computing and Networking (MobiCom '19)*, Los Cabos, Mexico, Oct. 2019, pp. 46:1–46:17.

[3] E. Kuznetsov, Y. Chen, and M. Zhao, "SecureFL: Privacy Preserving Federated Learning with SGX and TrustZone," in *Proc. 6th ACM/IEEE Symp. Edge Computing (SEC '21)*, San Jose, CA, USA, Dec. 2021, pp. 1–13.

[4] J. Guo, P. Pietzuch, A. Paverd, and K. Vaswani, "Trustworthy AI using Confidential Federated Learning," *ACM Queue*, vol. 22, no. 2, May 2024. [Online]. Available: https://queue.acm.org/detail.cfm?id=3665220

[5] E. Apsey, P. Rogers, M. O'Connor, and R. Nertney, "Confidential Computing on NVIDIA H100 GPUs for Secure and Trustworthy AI," *NVIDIA Technical Blog*, 03 Aug. 2023. [Online]. Available: https://developer.nvidia.com/blog/confidential-computing-on-h100-gpus-for-secure-and-trustworthy-ai/. [Accessed: 16-Jun-2025].

[6] P. Rogers and A. Delignat-Lavaud, "Hopper Confidential Computing: How it Works Under the Hood," in *NVIDIA GTC Spring 2023*, Mar. 2023. [Online]. Available: https://static.rainfocus.com/nvidia/gtcspring2023/sess/1666639437498001endS/supmat/S51709%20-%20Hopper%20Confidential%20Computing_%20How%20it%20Works%20under%20the%20Hood_1679465925191001GNep.pdf. Accessed: Jun. 18, 2025.

[7] Y. Yang, M. Sonji, and A. Jog, "Dissecting Performance Overheads of Confidential Computing on GPU-based Systems," in *Proc. IEEE Int. Symp. Performance Analysis of Systems and Software (ISPASS)*, Ghent, Belgium, May 2025, to appear.

[8] A. Mohan, M. Ye, H. Franke, M. Srivatsa, Z. Liu, and N. M. Gonzalez, "Securing AI inference in the cloud: Is CPU–GPU confidential computing ready?," in *Proc. IEEE 17th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2024, pp. 164–175.

[9] G. Dhanuskodi, S. Guha, V. Krishnan, A. Manjunatha, R. Nertney, M. O'Connor, and P. Rogers, "Creating the First Confidential GPUs," Commun. ACM, Practice, Jan. 8, 2024. [Online]. Available: https://cacm.acm.org/practice/creating-the-first-confidential-gpus/

[10] J. Hur,"sev-snp-gpu," GitHub repository, 2025. [Online]. Available: https://github.com/JaewonHur/sev-snp-gpu [Accessed: 16-Jun-2025].

[11] NVIDIA Corporation, "cuda-samples: bandwidthTest sample," GitHub repository, 2024. [Online]. Available: https://github.com/NVIDIA/cuda-samples/tree/main/Samples/1_Utilities/bandwidthTest. Accessed: Jun. 18, 2025.

[12] Advanced Micro Devices, Inc., "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More," White Paper, Jan. 2020. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf :contentReference[oaicite:0]index=0

[13] H. Uranami, A. Saiki, and K. Kimura, "Evaluation of GPGPU Data Transfer Overhead on a Secure VM," *IEICE Tech. Rep.*, vol. 2025-SLDM208, no. 14, pp. 1–6, Mar. 2025.

[14] Intel Corporation, "Intel Trust Domain Extensions (TDX) Whitepaper," Feb. 2022. [Online]. Available: file:///Users/yoshisato/Downloads/TDX-Whitepaper-February2022.pdf. Accessed: Jun. 18, 2025.

[15] Intel Corporation, "Intel® TDX Module 1.5 Base Architecture Specification," Intel Document #773614-001, Section 11.1: Virtualization Exception Handling, Aug. 2023.

[16] Intel Corporation, "Guest-Host-Communication Interface (GHCI) for Intel® Trust Domain Extensions (Intel® TDX)," White Paper, Version 1.5, Feb. 2022.

[17] "Intel Trust Domain Extensions (TDX)," The Linux Kernel Documentation, Section: Shared Memory Conversions. [Online]. Available: https://docs.kernel.org/arch/x86/tdx.html