Confidential LLM Inference: Performance and Cost Across CPU and GPU TEEs

Marcin Chrapek
ETH Zurich
Zurich, Switzerland
marcin.chrapek@inf.ethz.ch

Marcin Copik
ETH Zurich
Zurich, Switzerland

Etienne Mettaz
ETH Zurich
Zurich, Switzerland

Torsten Hoefler
ETH Zurich
Zurich, Switzerland

2025 IEEE International Symposium on Workload Characterization (IISWC) | 979-8-3315-4917-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/IISWC66894.2025.00017

***Abstract*—Large Language Models (LLMs) are increasingly deployed on converged Cloud and High-Performance Computing (HPC) infrastructure. However, as LLMs handle confidential inputs and are fine-tuned on costly, proprietary datasets, their heightened security requirements slow adoption in privacy-sensitive sectors such as healthcare and finance. We investigate methods to address this gap and propose Trusted Execution Environments (TEEs) as a solution for securing end-to-end LLM inference. We validate their practicality by evaluating these compute-intensive workloads entirely within CPU and GPU TEEs. On the CPU side, we conduct an in-depth study running full Llama2 inference pipelines (7B, 13B, 70B) inside Intel's TDX and SGX, accelerated by Advanced Matrix Extensions (AMX). We derive 12 insights, including that across various data types, batch sizes, and input lengths, CPU TEEs impose under 10% throughput and 20% latency overheads, further reduced by AMX. We run LLM inference on NVIDIA H100 Confidential Compute GPUs, contextualizing our CPU findings and observing throughput penalties of 4–8% that diminish as batch and input sizes grow. By comparing performance, cost, and security trade-offs, we show how CPU TEEs can be more cost-effective or secure than their GPU counterparts. To our knowledge, our work is the first to comprehensively demonstrate the performance and practicality of modern TEEs across both CPUs and GPUs for enabling confidential LLMs (cLLMs).**

***Index Terms*—Confidential LLMs; Trusted Execution Environments; Benchmarking; Inference; Performance Study**

I. Introduction

Large Language Models (LLMs) dominate the machine learning (ML) landscape [18], [91]. Exemplified by model families such as GPT [20], [64] and Llama [5], [41], [78], [79], they have become prevalent in industry and everyday life across a growing number of domains. LLMs achieve human-like capabilities on multimodal data [85] and have been applied to disciplines relying on *confidential* user information, including healthcare [71], finance [86], sentiment analysis [16], legal cases [29], and document translation [47]. Simultaneously, the ever-increasing size of LLMs has led to changes in their deployment strategies. LLMs ranging from billions to trillions of parameters necessitate state-of-the-art hardware to meet their performance demands, which is frequently provided by cloud service providers (CSPs).

However, deploying within the cloud carries security risks for LLMs that operate on expensive and confidential data. Figure 1 shows attacks that cloud providers, cluster administrators, and other tenants can carry out to leak model information and influence inference results. Data confidentiality and intellectual property (IP) theft pose critical threats to LLMs, for which the cost of engineering and obtaining datasets is substantial. With training and fine-tuning alone amounting to tens of millions of dollars [73], any security breach involving LLMs is increasingly costly for CSPs, model providers (e.g., MetaAI, OpenAI, financial or healthcare institutions), and end users.

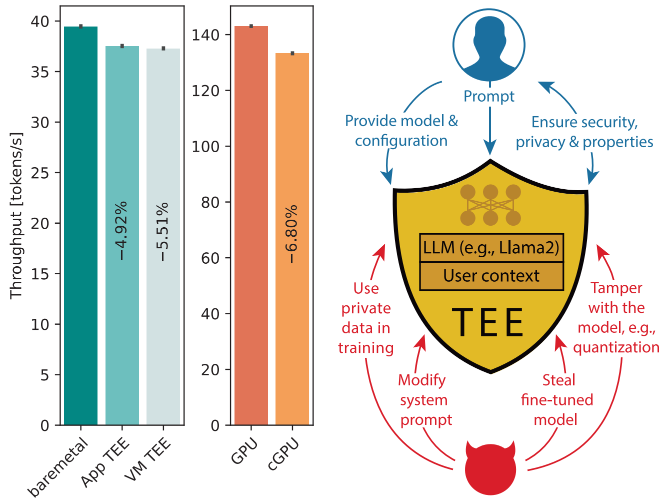


Fig. 1. Example attacks on LLMs that TEEs protect against and our performance results for Llama2 7B inference in two CPU TEEs: a Virtual Machine (VM, TDX) and an application-based (App, SGX) one, and a GPU TEE (cGPU).

While such threats might seem distant, they led to companies banning internal LLM use [8] and are tangible. For example, health records processed by a cloud-deployed LLM for insurance could be stolen and used maliciously. Even if leveraged solely for illicitly training another model, such a model might then be probed through public queries, reconstructing sensitive data, including names, addresses, Social Security numbers, and full medical histories [22], [66]. We also observe reports of backlash against leveraging user data for AI features [63] as more companies offer personalized AI (e.g., Meta's AI Studio or Adobe Creative).

The security community has addressed the issues of ML IP theft and data confidentiality by employing three primary

2835-2238/25/$31.00 ©2025 IEEE
DOI 10.1109/IISWC66894.2025.00017

approaches: model modifications (e.g., watermarking and user authentication [33], [46], [88]), cryptographic methods (e.g., homomorphic encryption [36]), and trusted execution environments [57]. We conduct an analysis of these techniques in Section II and show that TEEs currently provide the only viable method for protecting LLM inference. TEEs offer a practical balance between robust security properties, performance costs, and generalizability.

Our work focuses on quantifying usability and performance overheads of TEEs for protecting LLM inference by evaluating representative implementations of both CPU and GPU TEEs. In Section III, we start by evaluating the CPU side and conducting an in-depth study of Intel's Trust Domain Extensions (TDX) and Software Guard Extensions (SGX), representing common approaches to implementing TEEs: through virtual machines (VMs) and processes. We identify the best-performing frameworks and present the TEE performance overheads for throughput and latency in an end-to-end Llama2 (7B, 13B, and 70B) inference pipeline across various batch sizes, input lengths, and data types. Leveraging this compute-intensive workload, we derive 12 key insights on the performance of confidential LLM (cLLM) hosting, with practical guidelines for users and cloud providers. Our insights can be generalized to other TEE deployments and LLM systems. For example, we demonstrate how Advanced Matrix Extensions (AMX) directly result in lower overheads for TEEs. Figure 1 displays our example performance results, showing that TEEs for LLMs incur only 4-7% throughput reduction compared to overheads of up to 100s of percent for other applications [14], [27], [55].

In Section V-D, we present GPU results evaluated on NVIDIA's H100s that put our CPU results in perspective. We compare these two setups, considering cost, performance, and security. For example, we show that with AMX, CPU-based TEEs can be more cost-efficient than confidential NVIDIA H100 GPUs. Finally, in Section VI, we evaluate one of the most common LLM extensions: Retrieval Augmented Generation (RAG) [39]. We run full RAG pipelines, including Elasticsearch databases, in TEEs, and report their 7% overheads. We demonstrate how our lessons on CPU and GPU TEEs directly extend to these types of deployments. To our knowledge, our research is the first to comprehensively demonstrate the performance and practicality of modern TEEs across both CPUs and GPUs for enabling cLLMs. Our work can be replicated and evaluated seamlessly on other systems with the open-source implementation and configuration we release.

In summary, our contributions are:

1) Demonstrating how TEEs currently offer the only pragmatic solution for protecting LLM inference.
2) Characterizing performance of CPU TEEs (SGX, TDX) on Llama2 (7B/13B/70B), showing overheads of less than 10% for throughput and 20% for latency, identifying sources of performance degradation and optimal configurations.
3) Demonstrating how these relate to GPU TEEs by comparing with CPU TEEs in terms of cost-effectiveness, performance, and security.
4) Open-sourcing our configuration[1] and drawing 12 insights from empirical results, guiding efficient deployment and TEE system design.

II. Protection Mechanisms for LLM Inference

Three approaches can be used to protect LLM inference: machine learning (ML) methods, cryptographic methods such as Homomorphic Encryption (HE) and multiparty computation (MPC), and Confidential Computing (CC) [59].

ML methods: As noted in literature [88], current ML methods focus on post hoc detection of intellectual property (IP) theft in the form of model verification and passive protections, falling short in actively covering against model or data theft. Example approaches include using signatures embedded in the model with model theft verification using input/output pairs [49], passport [37] or backdoor [87] authentication, and watermarks in model output or weights used for ownership verification [19], [76].

While these protect against specific attacks, they do not provide exhaustive and measurable security properties. The cost of losing confidentiality or IP theft makes it challenging to rely only on them. Additionally, ML methods frequently require expensive retraining, alter the model's accuracy, fail to secure the confidentiality of user prompts [88], and cannot be combined together [75]. Cryptographic approaches, such as HE and MPC, address these issues through strong cryptographic protocols.

Cryptographic methods: HE allows conducting mathematical and logical operations on encrypted data without decrypting [13]. HE has been explored in the context of DNNs [33], [51], [84]. However, except for a few structured examples [21], [24], the state-of-the-art HE is not practical. HE operations on encrypted data can have up to 10,000x performance and size overheads, taking minutes to conduct simple MNIST [33] or RESNET [36] inference, and making LLM inference intangible. HE approaches also do not provide integrity protection. MPC is close to HE and has similar practicality issues, but involves multiple parties [82].

Confidential Computing: CC offers an alternative in the form of TEEs, using security primitives implemented in hardened hardware. Compared to HE and MPC, which rely on obscuring data and functions, TEEs offer a secure and isolated environment, frequently referred to as an *enclave*. Users can verify enclaves in a safe, hardware-enabled process called *attestation*. TEEs ensure the confidentiality and integrity of running programs and their data, protecting against external and privileged attackers, such as system administrators. TEEs achieve this by prohibiting access to or modification of the memory contents of running programs [70], including sensitive data like weights or user-confidential information.TEEs widely available on CSP platforms include CPU-based examples such as AMD's Secure Encrypted Virtualization-Secure Nested Paging (SEV-SNP) [45], Intel's SGX [28], [42], [54] and TDX [23], ARM's TrustZone [67] and CCA [53], and GPU-based examples such as NVIDIA's Confidential GPUs [62].

[1] github.com/spcl/confidential-llms-in-tees

Although TEEs do not provide the formal guarantees of HE or MPC, they still offer quantifiable defenses, particularly against integrity attacks that HE and MPC cannot address. Unlike many ML approaches, TEE protection mechanisms actively ensure enforcement of trust boundaries. However, performance and programmability are often cited as the two primary limitations of TEEs [14]. Because their security primitives lie on the critical path, TEEs incur non-negligible overhead. Nonetheless, as we show in our evaluation, TEE's overheads remain substantially lower than those imposed by HE schemes. Similarly, although TEEs require some security expertise, leveraging VM TEEs and frameworks like Gramine [81] eliminates the need for application modifications necessary for HE and ML methods.

> 🔑 **Insight 1:** TEEs offer a practical balance between security, performance, and programmability.

III. CPU TEEs

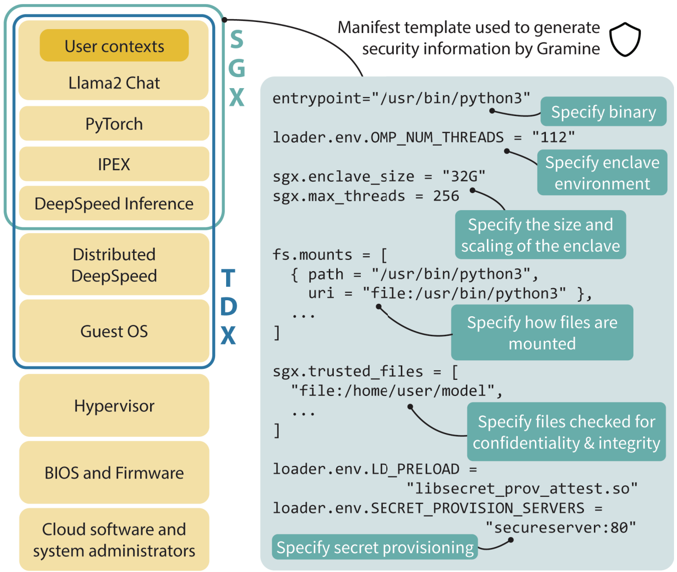
To investigate practical deployments, we limit ourselves to CPU TEEs offered by major CSPs. The options are limited to AMD and Intel since other TEEs, such as ones based on RISC-V [50] or ARM [67], are not widely available. We selected Intel's TEEs for two reasons. Firstly, they include support for AMX, an on-chip matrix operation hardware accelerator that introduces CPU-native support for formats such as brain-floating-points (`bfloat16`) and 8-bit integers (`int8`). AMX improves LLM performance 2-6x [61] (Figure 8), and we investigated whether these units also impact the performance of TEEs (Section IV-C). Secondly, they provide us with two common ways of implementing TEEs (VMs and processes) within the same system, covering other TEEs and enabling an apples-to-apples comparison without scaling performance results. For example, AMD's TEE stack relies on similar security mechanisms to Intel's TDX, resulting in close benchmark overheads [55].

A. Process-based TEEs: SGX

SGX programming model differentiates between SGX-protected and unprotected program sections. The former, located within an enclave, is safeguarded by SGX capabilities, while the latter is unsecured. SGX has two sources of overhead. First, data in the enclave is protected by memory encryption and integrity checks. Second, operations switching to the SGX unprotected program sections (e.g., IO such as reading a file) save SGX state and invalidate the caches.

SGX enclaves are frequently deployed on top of library operating systems (OSs) created for TEEs, such as Gramine [81] or Occlum [74]. These are lightweight layers between the host system and applications, intercepting any system calls to ensure they are conducted securely. These address some inconveniences of the original SGX software development kit (SDK), which required users to manually rewrite applications with secure and insecure sections.

In our study, we use the open-source Gramine [81] library OS that enables porting applications to SGX without significant code modifications. Unlike alternatives, it has lower requirements on the format of protected applications [74], is not proprietary [17], and is mature [72]. Gramine automatically applies integrity and confidentiality protections to storage, simplifies attestation, and transparently uses instructions for leaving and entering the SGX enclave during system calls. To increase performance, Gramine emulates some system calls without exiting the SGX enclave. However, if a given call is not implemented fully, it can result in considerable overhead. As we experienced firsthand, this can create a challenge while working with SGX, especially with complex workloads.


User contexts
Llama2 Chat
PyTorch
IPEX
DeepSpeed Inference
Distributed DeepSpeed
Guest OS
Hypervisor
BIOS and Firmware
Cloud software and system administrators
SGX
TDX
Manifest template used to generate security information by Gramine
Specify binary
Specify enclave environment
Specify the size and scaling of the enclave
Specify how files are mounted
Specify files checked for confidentiality & integrity
Specify secret provisioning

```
entrypoint="/usr/bin/python3"

loader.env.OMP_NUM_THREADS = "112"

sgx.enclave_size = "32G"
sgx.max_threads = 256


fs.mounts = [
  { path = "/usr/bin/python3",
    uri = "file:/usr/bin/python3" },
  ...
]

sgx.trusted_files = [
  "file:/home/user/model",
  ...
]

loader.env.LD_PRELOAD =
               "libsecret_prov_attest.so"
loader.env.SECRET_PROVISION_SERVERS =
                        "secureserver:80"
```

Fig. 2. Our software stack with the layers we protect in Intel TDX and SGX, and an extract from our Gramine manifest template file.

Gramine exposes its features via a Manifest file, which outlines the enclave size, the number of threads, the binary to be run, the files that can be trusted, and where to obtain the cryptographic decryption keys. Figure 2 shows an example excerpt from a Manifest file.

B. VM-based TEEs: TDX

TDX is a virtual machine (VM) based TEE that introduces security features using a hardened hardware-enabled kernel virtual machine (KVM) hypervisor. In the TDX security model, the entire VM is protected. This approach aligns well with the CSP virtualization trend and significantly simplifies development, eliminating the need for special functions when entering or exiting the enclave. TDX also runs programs within a standard Linux OS, such as Ubuntu, allowing for the easy execution of complex distributed AI frameworks, such as DeepSpeed [15], which we use. However, this convenience comes at the price of an increased attack surface. TDX requires trusting the entire VM OS and associated services, rather than just a minimal library OS, like in SGX. Using VMs also implies a virtualization performance tax, which can reach SGX's overheads as we demonstrate in Section III-D. Furthermore, some security aspects handled by frameworks, such as Gramine, are not performed automatically in TDX.

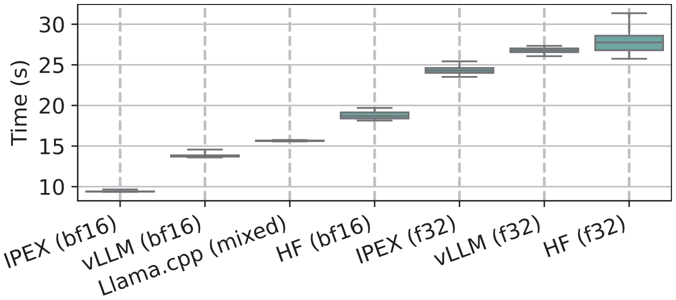


Fig. 3. Comparison of single-socket, bare metal wall CPU runtime on EMR1 of different backends and datatypes for Llama2 7B inference over 1024 input and 128 output tokens with beam and batch sizes equal to 1. HF is Hugging Face, bf16 is bfloat16, f32 is float32.

For example, users must protect the filesystem, e.g., by using Linux Unified Key Setup (LUKS) [38] full-disk encryption.

To use TDX, one must define a VM with a Quick Emulator (QEMU) command or a libvirt definition file. These specify hardware details, such as boot files, virtual-to-physical core mapping, and memory size, and result in a greater performance impact than enabling TDX (Section IV).

Insight 2: TDX is considerably easier to work with than SGX, especially for complex workloads.

C. Experimental setup

1) Hardware and software: We used two Emerald Rapid dual-socket Intel systems. First EMR1, a dual socket Intel Xeon® Gold 6530 ($2130 [3]), each with 32 cores, 16x32GiB 4800MHz DDR5 memory, Ubuntu 23.10, Python 3.10.12, PyTorch 2.2.0, transformers 4.35.2, Intel extension for PyTorch (IPEX) 2.2.0, and oneCCL PyTorch bindings 2.2.0. Second EMR2, a dual socket Intel Xeon® Platinum 8580 ($10710 [4]), each with 60 cores, 16x32GiB 4800MHz DDR5 memory, Ubuntu 24.04, Python 3.10.16, PyTorch 2.3.0, transformers 4.38.1, IPEX 2.3.100, and oneCCL PyTorch bindings 2.3.0.

2) Microbenchmark to select framework: To determine the best framework for inference on the CPU, we evaluated multiple popular options and assessed their performance across various data types using an example Llama2 7B LLM. We compared Hugging Face's transformers [83] (float32, bfloat16), vLLM [48] (float32, bfloat16), IPEX, and Llama.cpp [12] (mixed datatype). As Figure 3 shows, IPEX is considerably faster than all other frameworks, with the second vLLM being 50% slower and Hugging Face 100% slower. IPEX leverages AMX and its native `bfloat16` support to achieve the best performance. It also utilizes the oneAPI Collective Communications Library (oneCCL), which is fine-tuned for Intel's processors, making it a suitable choice for running across multiple NUMA domains.

Insight 3: Leveraging IPEX, and its AMX and oneCCL backends can double CPU inference performance.

3) Experiment details: We selected Llama2 [79] as a representative example of dense transformers. While subsequent iterations of the Llama family [5], [41] introduce models of different sizes, larger context windows, or mixtures of experts, they are fundamentally based on the same computational patterns. In this sense, Llama2 also represents well other dense transformer LLMs, such as GPT or OPT. This has been confirmed empirically by consistent performance patterns between these LLMs [61]. To verify that this is similar for TEEs, we also evaluated Llama3 8B, GPT-J 6B, Falcon 7B, Baichuan2 7B, and Qwen 7B, and found 3.1-13.1% overheads, in line with our Llama 7B results. We report user-perceived performance: throughput (tokens per second) and latency (time to receive next token). For latency, we measured the generation time for each token and its inverse for throughput. We run multiple generations for each experiment, measuring at least 1000 output tokens. We used two inference data types: `bfloat16` and `int8`. For the latter, we quantized the models. We evaluate four hardware configurations: the baseline represents results from a bare-metal machine, SGX from Gramine v1.7 running on SGX, VM from a raw VM without security features, and TDX from a TDX-enabled VM.

D. Single socket

We first establish baseline performance. Figure 4 shows the throughput (batch size = 6, beam size = 4) and the next token latency (batch size = 1, beam size = 1). The overhead of Gramine-SGX is between 4.80-6.15% while for TDX it is between 5.51-10.68%. TDX adds overhead of 3.02-7.01% over VM. The results for different data types show that `int8` generally achieves similar throughput to `bfloat16` but almost half the latency. While in SGX, the overheads for `int8` are similar to those for `bfloat16`, TDX shows considerable differences, where `int8` results are better in terms of throughput but worse in terms of latency. For throughput, lower memory movement due to the inference state in `int8` and the corresponding reduction in necessary address translations from guest to host memory results in lower overheads. For latency, memory access costs due to address translations and TEE memory protections are more pronounced when it is lower. All systems have a latency considerably below the average human reading speed of 200 ms/word (approximately 300 words per minute) [69], which forms a performance standard that LLMs should meet. As we plot per-token statistics, we noticed outliers for SGX and TDX, which we excluded in the violin plots using a `Z-score` > 3 ($\approx$0.64% of samples). As visible in the later plots, these do not contribute to the discussion but create considerable noise due to variability in memory encryption.

Insight 4: TDX and SGX have overheads as low as 4-10% for cLLM inference, preserving acceptable service performance.

The performance of SGX lies between that of a VM and TDX. In our deployment, SGX runs on bare metal, where the host OS has more privileges than a VM and exposes the hardware more directly. TDX, on the other hand, does not have direct access to specific hardware features and must access the underlying system through virtualization layers, such as guest address translations not present in SGX.

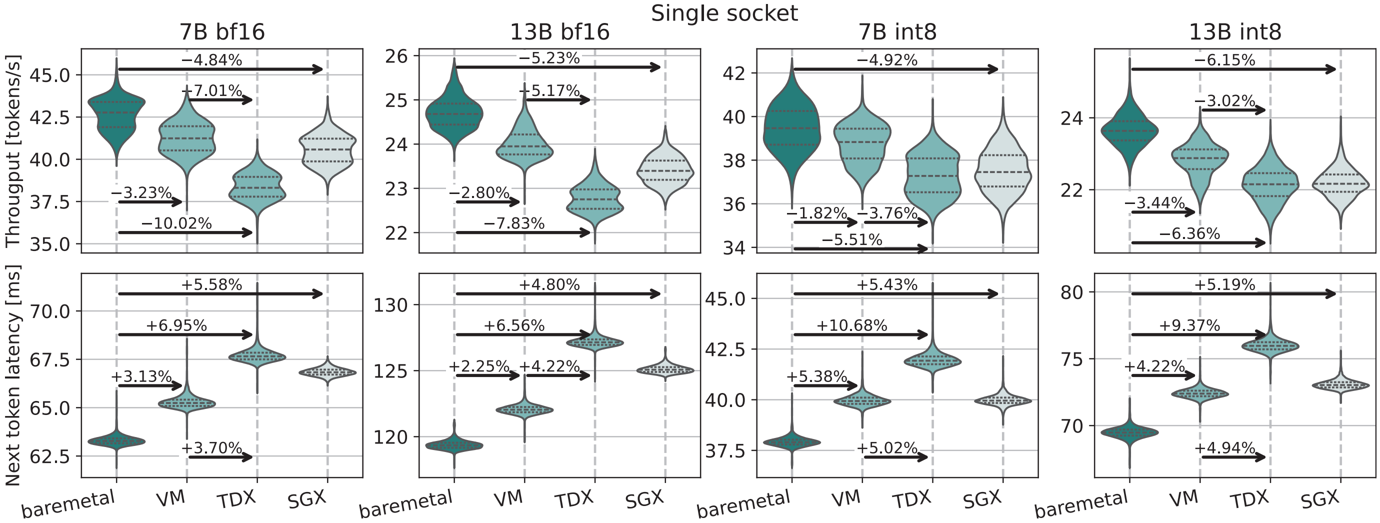


Fig. 4. TDX and SGX throughput and latency overheads stay within 4-10% for Llama2, and 1024 input, 128 output tokens on EMR1. A larger batch size implies increased latency and throughput as less data movement is required per token. Inputs batched are computed on each layer, and a combined result is forwarded to the next layer. Each layer has an increased latency over a single input but a decreased one over *N* separate inputs (increased throughput).

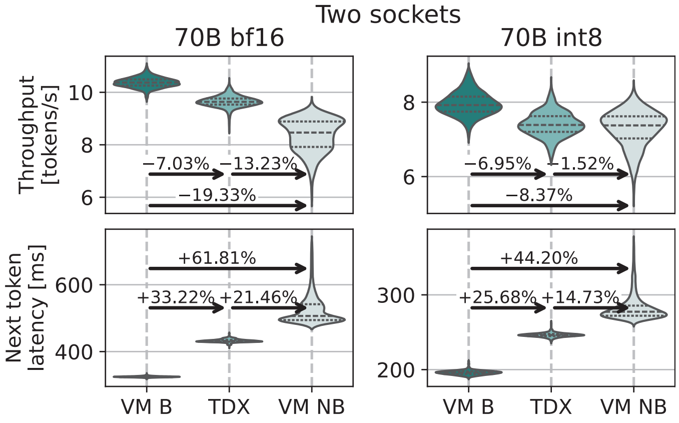


Fig. 5. The latency and throughput overheads of TDX over the VM backed with 2MB transparent huge pages (VM B) and VM backed with the same huge pages but without any NUMA binding (VM NB) on EMR1.

The results quantify this virtualization tax by showing that running in a VM has an overhead of 1.82-5.38%. The cost of security is similar for SGX and TDX, as the overheads of SGX over bare metal and TDX over VM are comparable.

🔑 **Insight 5:** Compared to SGX, TDX simplifies deployment but increases the trust boundary and pays a virtualization tax of 1-5%, making SGX more performant.

IV. Tuning CPU TEE overheads

Our investigation revealed three key areas to achieving acceptable performance within TEEs: appropriate TEE configuration, use of AMX, and optimizing memory efficiency.

A. Configuring TEEs to avoid performance traps

For SGX, we used the largest possible enclave page cache (EPC), which significantly influences overheads. EPC is a secure, SGX-exclusive, limited-size memory area that acts as a cache for encrypted enclave code and data. EPC enhances performance by minimizing costly paging to regular memory, which requires verification. Similarly, we observed higher performance without exposing the CPU core's second logical thread (hyperthread) to TDX. In its default configuration, PyTorch only executes on the first logical thread of a core, with hyperthreads introducing noise. We also identified more concerning limitations with non-uniform memory access (NUMA) and huge pages.

1) Multiple sockets: Figure 6 shows inference performance when deployed on two sockets. The performance overheads increase considerably, with TDX reporting an overhead of 12.11-23.81%. There are two reasons for such performance. First, the socket interconnect has a dedicated cryptographic unit [44], and any data moving between sockets must be encrypted and integrity-protected, which incurs a performance penalty on the critical path.

Second, TDX and SGX drivers lack working NUMA support. Figure 5 shows the performance of TDX when running on the 70B parameter model. This model is too large to fit into the memory of a single socket, and the 200ms service level is no longer upheld. We compare TDX performance to a VM with NUMA nodes bound in QEMU to the physical memory of two sockets (VM B) and non-bound (VM NB). While TDX is not as low-performing as VM NB, it has a considerable overhead compared to VM B, especially in terms of latency. We found that TDX's KVM driver does not adhere to the bindings that we provided.

We are not displaying the results of SGX as its overheads become prohibitively large, increasing up to 230%. While encryption on the socket interconnect reduces performance, such performance in SGX is predominantly due to a lack of proper support for NUMA. The memory is presented to the application as a single unified NUMA node, potentially resulting in the allocation of all memory on a single socket. While efforts have been made to optimize allocations to align

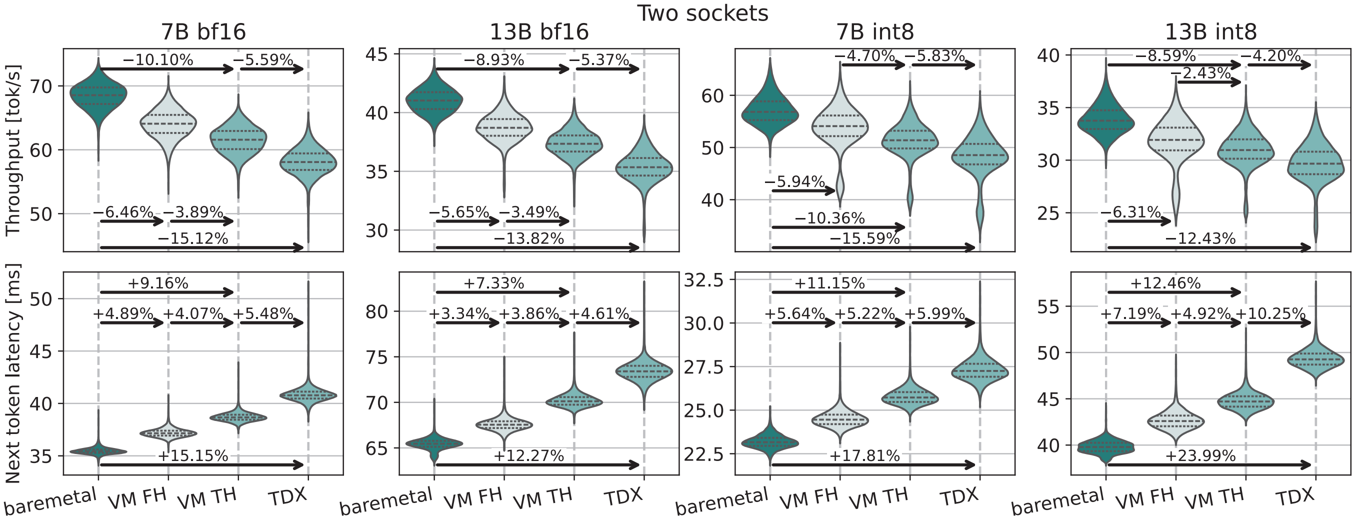


Fig. 6. The throughput and latency overheads for VM with full 1GB huge pages (VM FH), 2MB transparent huge pages (VM TH), and TDX on EMR1. The overheads of TDX over VM TH remain at 4-10%.

with the thread using the data [44], we have not found satisfactory performance of SGX in multiple sockets.

We also found that sub-NUMA clustering has a significant influence on both SGX and TDX. Sub-NUMA clustering (SNC) [60] in Intel CPUs divides a single socket into multiple NUMA domains, aiming at improving performance for ML workloads. TEE drivers also do not support sub-NUMA domains, resulting in inefficient memory placement. In our test runs, using sub-NUMA domains increased overhead by more than eight times, from approximately 5% to 42%. As a result, we disabled sub-NUMA clustering.

Insight 6: TDX and SGX do not properly support NUMA bindings, which leads to a considerably degraded performance, especially in the case of models that do not fit in the memory of a single socket.

2) Hugepages: For TDX, we also identified that it does not use 1GB huge pages [65], which increases the number of necessary translation lookaside buffer (TLB) accesses, worsening memory access latency. Figure 6 also shows the performance of different VM hugepage allocation strategies. VM FH uses preallocated 1GB hugepages, and VM TH uses 2MB transparent hugepages. TDX overheads over VM TH remain the same order of magnitude as in the single socket case. We found that TDX in the background uses transparent huge pages even if 1GB pages are provided. A larger data movement in the case of two NUMA nodes implies greater TLB pressure, manifesting in larger overheads of VM TH and TDX compared to VM FH and bare metal, for which huge pages matter less. The overhead of VM TH over VM FH quantifies the performance cost due to the lack of 1 GB hugepage support in TDX at 3.19–5.20%.

Insight 7: TDX uses self-allocated transparent hugepages and ignores manually reserved hugepages, which costs up to 5% of raw performance.

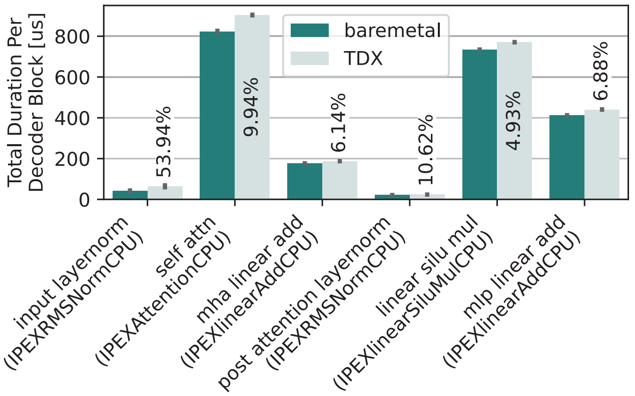


Fig. 7. The duration and TDX overhead of each decoder block layer for Llama7B on a single socket of EMR2.

B. Per-block overheads

To better understand the sources of overhead, we traced the single-socket inference of 128 in/out tokens for a batch size of 4 for TDX. We then parsed the traces to measure the time of each inference layer. We observed that decoder blocks take 99.9% of the time, with the remainder devoted to embedding and final normalization. Figure 7 shows the duration and overheads for each decoder block layer. The most significant overheads are incurred in input and post-attention layer norms. However, these have large relative noises and form only 3% of the total block time. The most considerable cost in raw performance is incurred in self-attention and linear SiLU multiplication. Given that these have a considerable data movement [43], it is clear that memory encryption is a major contributor to the overheads. The time these take is impacted by the arithmetic intensity, influenced by solutions such as AMX and operational parameters such as batch and input sizes. These two parameters also considerably impact the exact relative durations we have shown above. As we increased

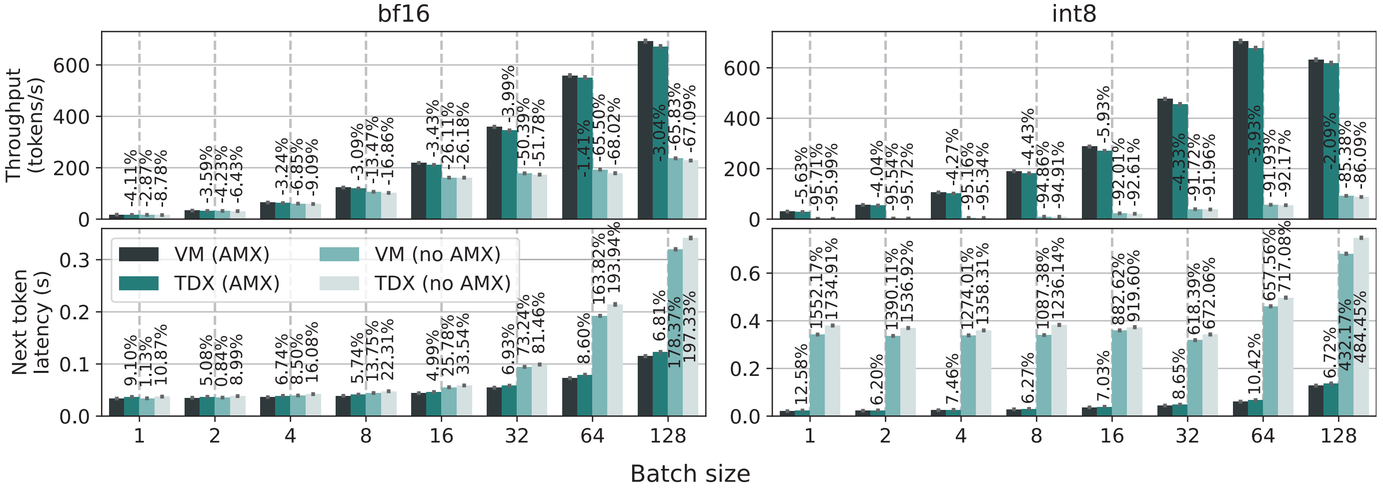


Fig. 8. Comparison of performance between AMX and no-AMX systems as we scale the batch size for Llama2 7B, with 128 in and out tokens and beam size equal to one on EMR2. The overheads are relative to VM running AMX. We show the best performing setups: latency on two sockets, throughput on one.

the batch and input sizes, we observed that self-attention and linear SiLU remain the most significant contributors to overall block time, with self-attention dominating even more.

C. Use of AMX

As shown in Section III-C, using IPEX, which leverages AMX, has a significant impact on inference performance. However, what we found is that AMX also minimizes TDX overheads. For further experiments, we focus solely on TDX, which performs worse than SGX, forming a lower bound on performance. However, it is easier to work with, especially for experiments that disable AMX, limit the number of cores, or run RAG pipelines. All VMs henceforth use 1GB hugepages.

Figure 8 investigates the benefits of AMX across batch sizes, against a setup running IPEX without AMX. In the case of `bfloat16`, AMX initially provides a slight advantage of 1-4%, which increases to hundreds of percent with larger batch sizes (more compute). AMX not only significantly influences raw performance but also reduces the overheads of TDX, lowering them by up to 30% for latency and up to 2% for throughput. As latency results are measured on two sockets, lower NUMA traffic caused by AMX explains these benefits. Importantly, we also observed up to 96% of overhead in throughput and 1700% in latency for `int8`. Such low performance occurs because the model quantization is fine-tuned for AMX, and there is a lack of AVX implementation for `int8` in IPEX.

Insight 8: AMX lowers TDX overheads, accelerates workloads up to 2.6x, and enables quantized inference.

D. Efficient use of memory

The final overheads we observed include memory protection costs, which are influenced by the amount of paging and the application's arithmetic intensity. We optimized the former by using TCMalloc [34], which reduces the memory pressure. For the latter, we used an OpenMP [30] version suitable for Intel processors. However, the choice of operational parameters, such as batch and input sizes, has a greater impact.

1) Batch size scaling: Figure 9 shows the results of varying the batch size. As it is scaled, we expect more algorithmic

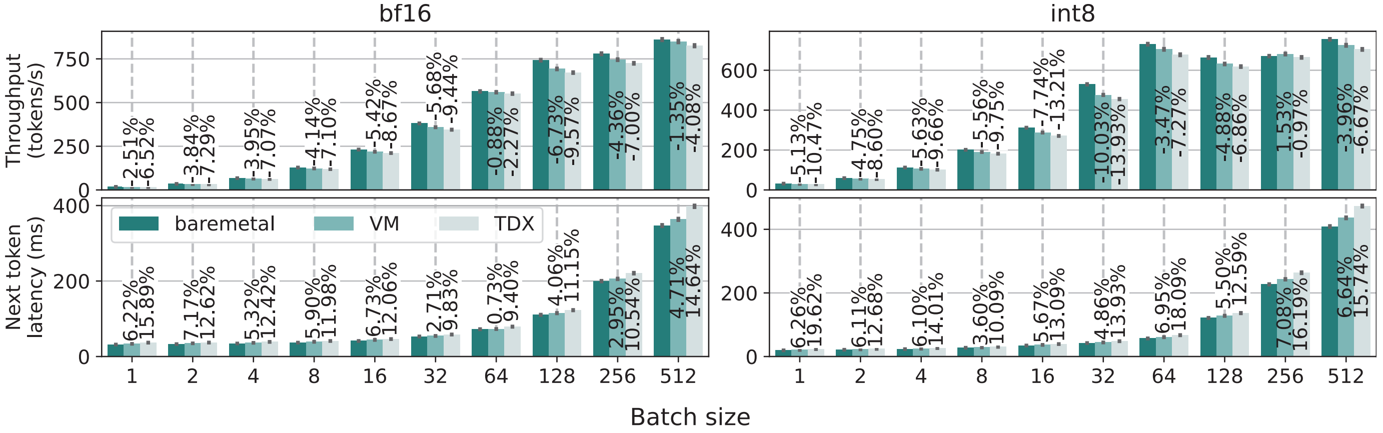


Fig. 9. Comparison of next token latency and throughput as we scale the batch size with 128 in and out tokens and beam size of one on EMR2. Performance overheads are shown relative to bare metal. Latency is measured on two sockets, while throughput is measured on a single socket. Throughput for two sockets equals twice the shown values.

intensity, which lowers TDX's overhead stemming from memory encryptions. This is precisely what we observe. For `int8`, the workload saturates the throughput at batch size 64 when the overheads drop from 9-11% to 6% or less. `bfloat16` also achieves throughput saturation, but around a batch size of 512. This is also when the overheads drop from 7-10% to 4-7%. From a latency perspective, we do not observe such a strong correlation, which is due to the overhead of socket interconnect data movement that also increases alongside algorithmic intensity. A batch size of 64 achieves the best performance for `bfloat16` throughput, when the overheads drop to 2%. As this marks the inflection point for bare metal performance, we evaluate it across different input sizes.

2) Input size scaling: Figure 10 shows the throughput performance against the input size. We observe that the overhead of TDX decreases as the input size increases, both for `int8` and `bfloat16`, until it reaches 2048 tokens. The overhead variability stems from the interplay of caches and AMX. As we initially increase the input size, we benefit from the workload saturating the AMX units and becoming more compute-bound, similarly to the batching case. However, as we increase the input size, the KV cache size per new token also grows. Eventually, it reaches the point where each token causes a considerable cache miss rate, making the workload memory-bound. We observe increased overheads for both TDX and VM, as this also leads to TLB misses. At this regime, we achieve overheads similar to smaller batch sizes.

Insight 9: TDX has the lowest overhead when the workload is compute-bound.

V. GPU TEEs

To put our CPU results in perspective, we also investigate cGPUs. At the time of writing, the GPU-based TEE introduced by NVIDIA in the Hopper architecture is the only accelerated TEE solution entering the space on a large scale. H100s with CC enabled are available only in production mode at Azure [1] and GCP [7]. Their successors, B100s, are currently not available in any CSP in the CC configuration.

A. NVIDIA Confidential GPUs

cGPUs require a host CPU TEE, enabling GPU attestation. Users can run their kernels on cGPUs without any changes to existing CUDA applications. All command buffers, kernels, and data transfers over PCIe between CPU and GPU are encrypted and authenticated via a bounce buffer. This prevents hypervisor or physical attackers from accessing sensitive information. These transfers, together with an additional kernel invocation latency, are the main costs of the current cGPUs. To avoid the PCIe overhead, solutions such as PCIe IDE need to be used [2]. While PCIe transfers are protected, the HBM memory of H100s is not. Additionally, the NVLINK communication is unprotected when combining multiple H100s, requiring secure communication through the host. The B100s resolve the main security issues of H100s and introduce HBM memory and NVLINK encryption. While B100s address these issues, their availability in CC configurations makes it challenging to evaluate the costs of these protections.

B. Experimental setup

We used an H100 NVL GPU with 94 GB of memory (~$30,000 [6]) rented from Azure (confidential `NCCads_H100_v5` and non-confidential `NCads_H100_v5`), with a 40-virtual CPU (vCPU) AMD EPYC 9V84 host and 320 GiB memory. We deployed Ubuntu 24.04 and leveraged vLLM [48] version 0.8.5 as an optimized inference framework. As our machine is rented, we do not have access to bare metal and present the results for raw and Confidential GPUs (cGPU).

C. Batch and input size scaling

Figure 11 shows the performance of GPUs for Llama2 7B for different batch sizes and input lengths. cGPUs exhibit similar performance to CPU TEEs, albeit with lower noise. This is an expected behavior since GPUs do not have encrypted memory on the critical path. As both batch size and input length increase, the cGPU performance improves, primarily due to increased arithmetic intensity. Since the share of time spent on setup remains roughly the same (including overhead-inducing kernel invocations and data transfers from the CPU), the overheads naturally decrease. While for inference, the data transfer is minimal, for workloads such as LLM training, it is large. Solutions such as TDX Connect [23] and SEV IO [9] are in development to address these overheads.

Insight 10: GPU TEEs achieve less than 10% overheads, which decreases with larger batch and input sizes.

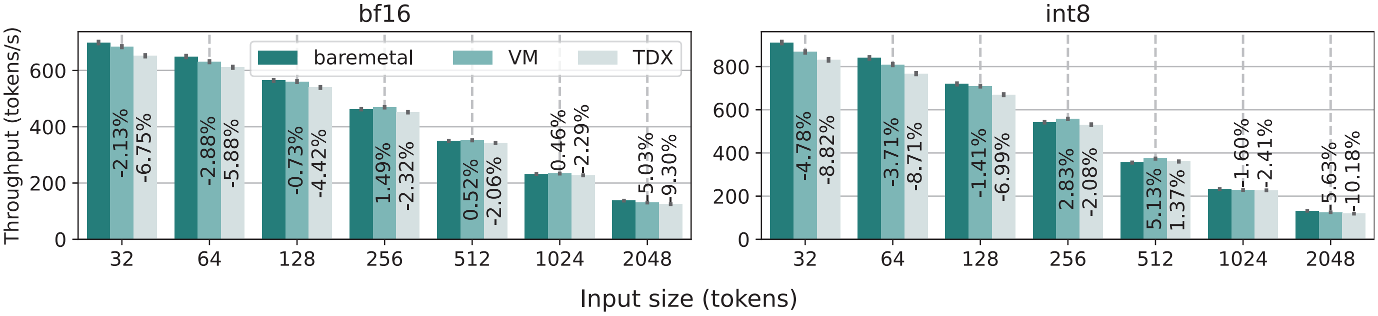


Fig. 10. Comparison of generation throughput as we scale the input size for Llama2 7B on a single socket, with 128 out tokens, beam size 1, batch size 64, on EMR2. The overheads are relative to bare metal.

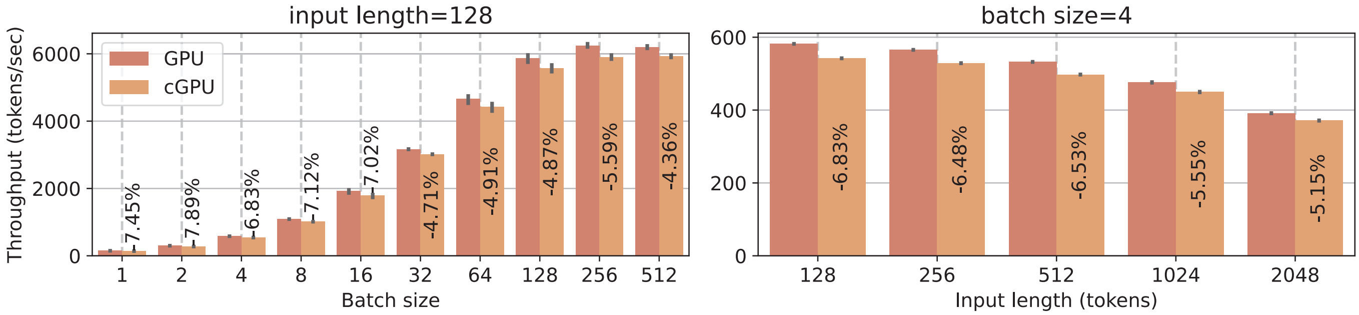


Fig. 11. GPU throughput as a function of batch and input sizes. As both increase, the overheads are minimized, and oscillate between 7.5% and 4.4%

D. Comparing CPUs and GPUs

1) Hybrid setups: Results in Section V-C indicate that the GPU has a much better raw performance. This occurs as long as the model can be entirely fitted on the GPU. Prior research has shown [61] that if parts of the model need to be offloaded to the host memory, the AMX-accelerated CPUs outperform GPUs. This is even more so the case for confidential computing, as any data movement between CPU and GPU is more expensive, due to the cost of encrypting the PCIe bounce buffer. We demonstrate that in the case of confidential computing, two additional scenarios arise in which CPUs outperform current GPUs.

2) Resource efficiency: Figure 12 shows the throughput across different batch sizes (columns) and numbers of CPU cores used during inference. The results indicate that the workload remains compute-bound until 32 cores, after which it becomes memory-bound, suggesting minimal performance gain above this number of cores. Similarly to prior plots, a batch size of 64 has the lowest TDX overheads.

Additionally, Figure 12 shows the cost of inference of 1 million tokens. To evaluate the cost of running different setups, we used spot prices offered by Google Cloud Platform (GCP) for the same machine type deployed in the US East 1 region. As GCP allows users to select the number of vCPU cores and the amount of memory separately, we assumed 128 GB of memory, which we found to be sufficient for deploying Llama2 7B in all the shown cases. We then scaled the number of vCPUs, keeping the memory size constant. Memory initially dominates the cost of renting, as it is fixed regardless of the number of CPU cores used. As we add cores, the performance increases, lowering the price per million tokens, which starts climbing back to 32 cores when the throughput plateau is reached. As we increase the batch size, the computational needs increase, making the larger machines more economical. For example, at a batch size of 128, 32 cores become optimal. As this workload becomes memory-bound easily, renting an almost 2x cheaper Sapphire Rapid performing up to 40% worse [35], provides an even more affordable alternative.

We marked the cGPU cost-effectiveness with an orange line in Figure 12. While more performant, GPUs also have a significantly higher price per hour, resulting in cGPUs being up to 100% more expensive. We observe that as the batch size increases, the advantage of CPU TEEs slowly fades, until it reaches a batch size of 128, at which point they equalize. Such behavior is expected as GPUs become more efficient with larger batch sizes [68]. LLM queries with low computational intensity are especially more cost-efficient when using TEEs. Currently, NVIDIA supports CC only on H100 and B100

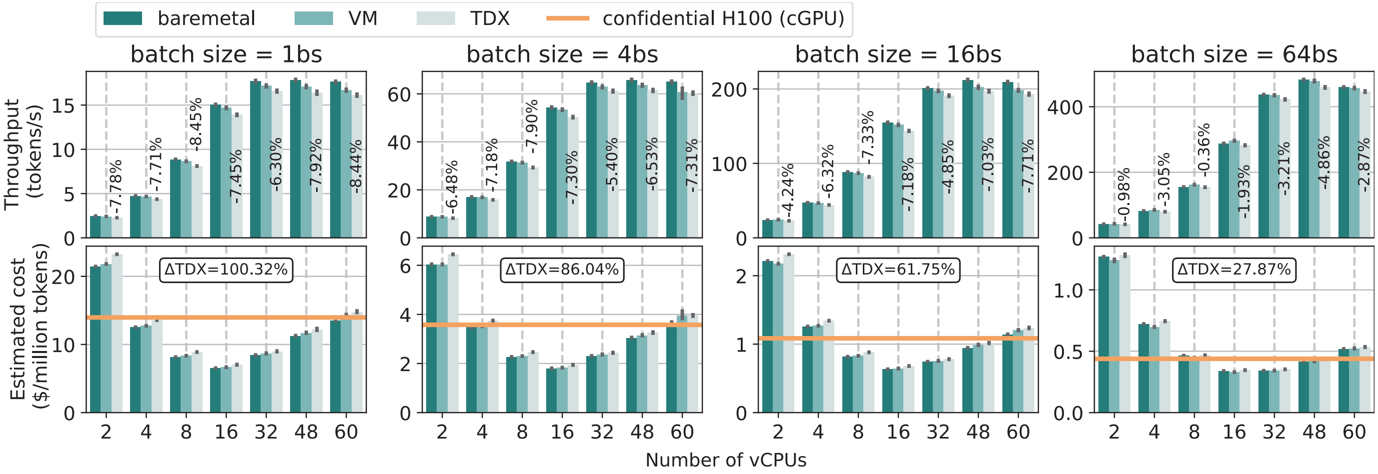


Fig. 12. vCPU scaling and cost of generating on EMR2. Generation throughput includes the first token latency, measured over 128 in and out tokens on a single socket for bfloat16. The throughput overheads are with respect to bare metal, and the cost of overheads of TDX with respect to GPU.

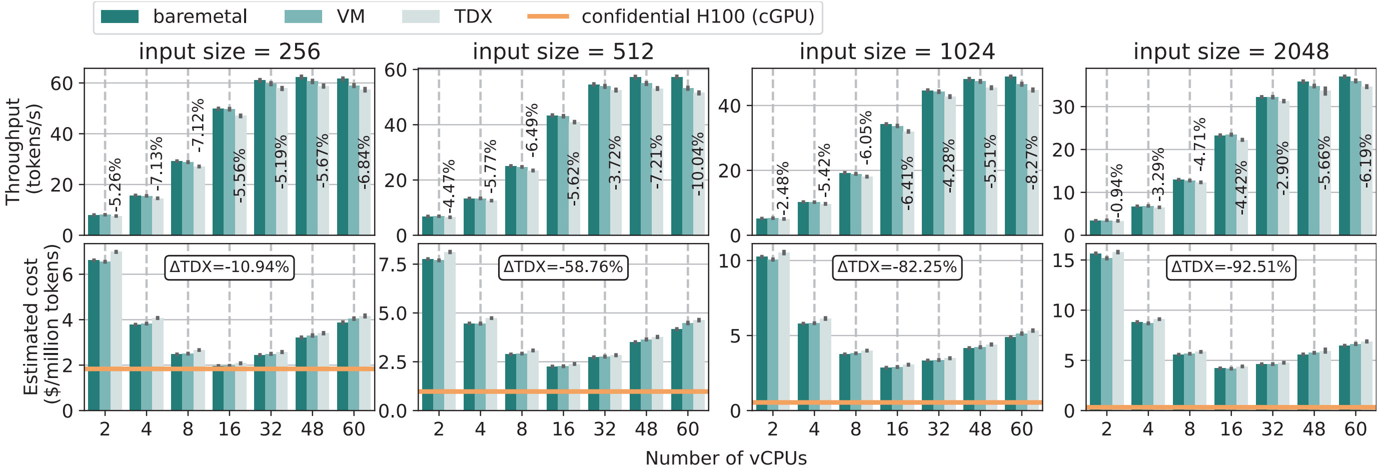


Fig. 13. vCPU scaling and cost of generating tokens on EMR2. Throughput includes the first token latency, measured over 128 out tokens, batch size 4, on a single socket for bfloat16. The throughput overheads are with respect to bare metal, and the cost of cGPU overheads with respect to TDX.

systems, excluding MIG [52] and older or less powerful GPUs, such as the A100, which are used to optimize cost efficiency.

To verify that CPU TEEs eventually lose their advantage when compute requirements are sufficient, we also evaluate performance with varying input sizes for a batch size of 4. As results in Figure 13 reveal, from the cost perspective, CPU TEEs are considerably more sensitive to input size than cGPUs. The first batch size for which CPU TEEs are uncompetitive is 128, losing the 100% resource advantage of batch size 1. However, we only needed to double the input size to achieve a similar reduction in gains, from 86% to -10%. As the attention part of the model grows quadratically with the input size, it implies a greater impact on compute requirements as compared to only linear increases for batch size.

3) Security: While CPU TEEs perform worse than cGPUs with larger input sizes, they have one more advantage: security. CPU TEEs are more mature, and their security model is stricter than cGPUs. H100s do not encrypt their HBM memory [31], compared to CPUs that do. While in CPU-based systems, communication between different sockets is transparently encrypted, interconnects such as PCIe and NVLINK do not yet have this feature [31], which limits inter-accelerator communication to go through the host. This is crucial for larger models that do not fit on a single GPU. While B100s address these issues, we expect that they will add a non-negligible overhead to H100s' results, since we identified memory encryption as a significant cost in CPUs.

4) Scaling models: We compared the CPUs (fitting 70B parameters) to a single GPU (fitting 30B parameters). Scale-up of confidential H100s is costly due to the aforementioned security concerns. Similarly, scaling out through combining single-GPU VMs is currently inefficient. As the cGPU instances do not support RDMA and GPUdirect, all data is transferred through the CPU, capping throughput at 3GB/s (considerably lower than the non-confidential 40GB/s) [89]. This is costly for throughput-hungry patterns such as pipeline parallelism and tensor parallelism. We expect this to lower the advantage of GPUs over CPUs. A network protection scheme, such as IPsec, is required on top of both CPUs and GPUs, which also introduces an overhead of up to 90% [25].

Insight 11: For strictest security workloads, and relatively small LLMs such as Llama2 7B, where H100 GPUs would be unsaturated (e.g., small batch or input sizes), CPU TEEs offer a pragmatic way to secure inference.

VI. Moving to RAG

RAG is a practical showcase of our insights. RAG is an extension of LLMs, enabling them to retrieve documents that match queries. RAG embeds documents in an index, which is then searched during inference for closest matches in a process called retrieval. For example, the Best Matching 25 (BM25) is a classic retrieval model that ranks documents by keywords. Reranked BM25 first retrieves BM25 and then reranks it using a cross-encoder. For both, an Elasticsearch database [10] is typically used to store the documents. RAG can also involve LLMs such as SBERT, which encodes queries and documents into dense vectors using a pre-trained Sentence-BERT encoder and ranks matches based on cosine similarity. We evaluate the performance of RAG using these three methods in BEIR [77], running them and an Elasticsearch database entirely within TDX. Figure 14 shows that even though the RAG workload, such as BM25 ranking, differs from a normal LLM inference, our results display a similar level of overhead. We observe 6-7% degradation for TDX, suggesting CPU TEEs might also be used for these purposes without significant performance impacts. Additionally, knowing that LLM RAG is conducted frequently with a batch size of one and for small models such as SBERT, we can leverage Insight 11 to deduce that CPU TEEs might be more cost-efficient than cGPUs.

Insight 12: Performance of entire RAG pipeline in TDX achieves similar overheads to the LLM inference.

VII. Related work

TEEs have been investigated in the past for protecting ML models [57]. Yet, most of these approaches offload only

		System	Intel SGX (process TEE)	Intel TDX (VM TEE)	H100 cGPU (GPU TEE)
Security	Hardware	Memory	■	■	□ (HBM unencrypted)
		Scale-up	■	■	◧ (NVLINK unprotected)
	Software	App	■	■	■
		OS	◧ (libOS)	■	■
		VM	□	■	■
Performance	Overhead	Single resource	~4-5%	~5–10%	~4–8%
	Parameters influencing overheads	Batch size↑	↓	↓	↓
		Input size↑	↓↑	↓↑	↓
		AMX	↓	↓	-
		Scale-up	↑↑	↑	↑↑
	Sources of overheads		EPC paging, enclave exits, memory, NUMA	Virtualization tax, hugepages, memory, NUMA	PCIe transfers, kernel launch
Cost	Development		◧	■	■
	Resource efficiency	Small inputs/batches	■	■	◧
		Large inputs/batches	◧	◧	■

TABLE I
The summary of evaluated systems and the insights. ■ indicates full/good, ◧ partial, and □ no support. ↓ indicates decreasing, ↑ increasing, ↑↑ increasing considerably more than ↑, and ↓↑ first decreasing, then increasing overheads.

parts of the models to TEEs, providing weaker notions of security and citing low TEE performance as the reason. For example, Slalom [80] offloads linear layers to the GPU with a probabilistic algorithm guaranteeing some security. Furthermore, none of these works explored LLMs, focusing instead on simpler models due to the extensive model changes. In contrast, we run an entire LLM inference pipeline in TEEs, demonstrating their practicality for protecting LLMs.

Some performance studies have been conducted on SGX [14], [32], [40], [56], [90] and TDX [11], [55]. These focus on quantifying the overheads of the underlying primitive operations, such as memory overheads, and the performance of certain applications. However, none address workloads as compute-intensive as LLMs. Some works that demonstrate secure LLM inference [26] focus more on security, missing the depth and key deployment insights, such as AMX performance improvements, scalability, and cost considerations. Similarly, GPUs have been studied for their sources of overhead [58], [89]. However, these outline overheads considerably larger than ours, or do not show raw LLM performance. Additionally, none compares GPU TEEs to CPU TEEs, thereby failing to display the full spectrum of practical deployments.

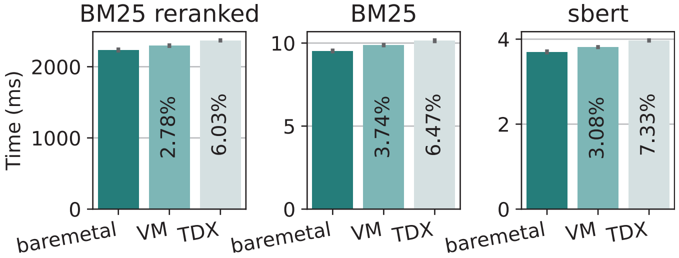


Fig. 14. Comparison of mean evaluation time for RAG systems on EMR2.

VIII. Conclusions

We investigated several methods for protecting LLM deployments and discussed how TEEs yield a practical balance between security, performance, and programmability. We demonstrated the viability of securing LLMs with TEEs by running an inference pipeline on top of Intel's TDX and SGX, as well as NVIDIA's H100s. We conducted a thorough study of the performance of TEEs in these workloads, identifying the best frameworks, sources of overheads, and optimal operating points. We shared 12 key insights, showing, among others, that CPU TEEs have NUMA and hugepages issues, and how AMX helps improve their performance. We have also compared CPU and GPU TEEs in terms of performance, cost-efficiency, and security. Finally, we applied our lessons to a RAG pipeline within a TEE, demonstrating its performance. Table I shows the summary of our investigation. Our results show that TEEs impose a manageable performance overhead on LLM pipelines, demonstrating that TEEs represent a viable solution for protecting LLM inference, positioning them as a cornerstone for future confidential AI deployments.

Acknowledgment

This research was conducted as part of the "UrbanTwin: An urban digital twin for climate action: Assessing policies and solutions for energy, water and infrastructure" project, funded by ETH-Domain Joint Initiative program in the Strategic Area Energy, Climate and Sustainable Environment, with additional support from Intel Corporation. We thank Intel for providing hardware resources, Cory Cornelius, Anjo Vahldiek-Oberwagner, Marcin Spoczynski, Scott Constable and Mona Vij for their valuable feedback, and Madlen Koblinger for assisting with the design of figures.

References

[1] "Announcing Azure confidential VMs with NVIDIA H100 Tensor Core GPUs in Preview," https://techcommunity.microsoft.com/t5/azure-confidential-computing/announcing-azure-confidential-vms-with-nvidia-h100-tensor-core/ba-p/3975389.

[2] "IDE and TDISP: An Overview of PCIe® Technology Security Features | PCI-SIG," https://pcisig.com/blog/ide-and-tdisp-overview-pcie%C2%AE-technology-security-features.

[3] "Intel® Xeon® Gold 6530 Processor (160M Cache, 2.10 GHz) - Product Specifications," https://www.intel.com/content/www/us/en/products/sku/237249/intel-xeon-gold-6530-processor-160m-cache-2-10-ghz/specifications.html.

[4] "Intel® Xeon® Platinum 8580 Processor (300M Cache, 2.00 GHz) - Product Specifications," https://www.intel.com/content/www/us/en/products/sku/237250/intel-xeon-platinum-8580-processor-300m-cache-2-00-ghz/specifications.html.

[5] "The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation," https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

[6] "NVIDIA H100 NVL 94GB | ASA Computers," https://www.asacomputers.com/nvidia-h100-nvl-94gb-graphics-card.html.

[7] "Privacy-preserving Confidential Computing now on even more machines," https://cloud.google.com/blog/products/identity-security/privacy-preserving-confidential-computing-now-on-even-more-machines.

[8] "Samsung Bans Generative AI Use by Staff After ChatGPT Data Leak," *Bloomberg.com*, May 2023.

[9] "SEV-TIO Firmware Interface Specification," Tech. Rep., 2023.

[10] "Elastic/elasticsearch," elastic, May 2025.

[11] "An experimental evaluation of TEE technology: Benchmarking transparent approaches based on SGX, SEV, and TDX," *Computers & Security*, vol. 154, p. 104457, Jul. 2025.

[12] "Ggml-org/llama.cpp," ggml, May 2025.

[13] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A Survey on Homomorphic Encryption Schemes: Theory and Implementation," *ACM Computing Surveys*, vol. 51, no. 4, pp. 79:1–79:35, Jul. 2018.

[14] A. Akram, A. Giannakou, V. Akella, J. Lowe-Power, and S. Peisert, "Performance Analysis of Scientific Computing Workloads on General Purpose TEEs," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2021, pp. 1066–1076.

[15] R. Y. Aminabadi, S. Rajbhandari, A. A. Awan, C. Li, D. Li, E. Zheng, O. Ruwase, S. Smith, M. Zhang, J. Rasley, and Y. He, "DeepSpeed-Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale," in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2022, pp. 1–15.

[16] D. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," Aug. 2019.

[17] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell, D. Goltzsche, D. Eyers, R. Kapitza, P. Pietzuch, and C. Fetzer, "{SCONE}: Secure Linux Containers with Intel {SGX}," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 689–703.

[18] M. Awais, M. Naseer, S. Khan, R. M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang, and F. S. Khan, "Foundational Models Defining a New Era in Vision: A Survey and Outlook," Jul. 2023.

[19] F. Boenisch, "A Systematic Review on Model Watermarking for Neural Networks," *Frontiers in Big Data*, vol. 4, p. 729663, Nov. 2021.

[20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," Jul. 2020.

[21] L. Burkhalter, A. Hithnawi, A. Viand, H. Shafagh, and S. Ratnasamy, "{TimeCrypt}: Encrypted Data Stream Processing at Scale with Cryptographic Access Control," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 835–850.

[22] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel, "Extracting Training Data from Large Language Models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.

[23] P.-C. Cheng, W. Ozga, E. Valdez, S. Ahmed, Z. Gu, H. Jamjoom, H. Franke, and J. Bottomley, "Intel TDX Demystified: A Top-Down Approach," *ACM Computing Surveys*, Mar. 2024.

[24] M. Chrapek, M. Khalilov, and T. Hoefler, "HEAR: Homomorphically Encrypted Allreduce," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '23. New York, NY, USA: Association for Computing Machinery, Nov. 2023, pp. 1–17.

[25] M. Chrapek, S. Shen, P. Iff, T. Chen, M. Khalilov, M. Copik, M. Besta, and T. Hoefler, "Secperf: Demystifying the cost of confidential compute," 2025.

[26] M. Chrapek, A. Vahldiek-Oberwagner, M. Spoczynski, S. Constable, M. Vij, and T. Hoefler, "Fortify Your Foundations: Practical Privacy and Security for Foundation Model Deployments In The Cloud," Oct. 2024.

[27] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "An experimental evaluation of TEE technology: Benchmarking transparent approaches based on SGX, SEV, and TDX," *Computers & Security*, vol. 154, p. 104457, Jul. 2025.

[28] V. Costan and S. Devadas, "Intel SGX Explained," 2016.

[29] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, "ChatLaw: Open-Source Legal Large Language Model with Integrated External Knowledge Bases," Jun. 2023.

[30] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, Jan. 1998.

[31] G. Dhanuskodi, S. Guha, V. Krishnan, A. Manjunatha, M. O'Connor, R. Nertney, and P. Rogers, "Creating the First Confidential GPUs: The team at NVIDIA brings confidentiality and integrity to user code and data for accelerated computing." *Queue*, vol. 21, no. 4, pp. Pages 40:68–Pages 40:93, Sep. 2023.

[32] T. Dinh Ngoc, B. Bui, S. Bitchebe, A. Tchana, V. Schiavoni, P. Felber, and D. Hagimont, "Everything You Should Know About Intel SGX Performance on Virtualized Systems," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 1, pp. 5:1–5:21, Mar. 2019.

[33] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy."

[34] D. Durner, V. Leis, and T. Neumann, "On the Impact of Memory Allocation on High-Performance Query Processing," in *Proceedings of the 15th International Workshop on Data Management on New Hardware*, ser. DaMoN'19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 1–3.

[35] D. Eadline, "Intel Won't Have a Xeon Max Chip with New Emerald Rapids CPU," https://www.hpcwire.com/2023/12/14/intel-wont-have-a-xeon-max-chip-with-new-emerald-rapids-cpu/, Dec. 2023.

[36] A. Ebel, K. Garimella, and B. Reagen, "Orion: A Fully Homomorphic Encryption Framework for Deep Learning," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '25. New York, NY, USA: Association for Computing Machinery, Mar. 2025, pp. 734–749.

[37] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking Deep Neural Network Ownership Verification: Embedding Passports to Defeat Ambiguity Attacks," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[38] C. Fruhwirth, "LUKS on-disk format specification version 1.2." 2011.

[39] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," Mar. 2024.

[40] A. T. Gjerdrum, R. Pettersen, H. D. Johansen, and D. Johansen, "Performance of Trusted Computing in Cloud Infrastructures with Intel SGX," in *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, ser. CLOSER 2017. Setubal, PRT: SCITEPRESS - Science and Technology Publications, Lda, Apr. 2017, pp. 696–703.

[41] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz,

D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng, O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma, "The Llama 3 Herd of Models," Nov. 2024.

[42] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, "Using innovative instructions to create trustworthy software solutions," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '13. New York, NY, USA: Association for Computing Machinery, Jun. 2013, p. 1.

[43] A. Ivanov, N. Dryden, T. Ben-Nun, S. Li, and T. Hoefler, "Data Movement Is All You Need: A Case Study on Optimizing Transformers," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 711–732, Mar. 2021.

[44] S. Johnson, R. Makaram, A. Santoni, and V. Scarlata, "Supporting intel® SGX on multi-socket platforms," Intel Corporation, Tech. Rep. 843058, Dec. 2024.

[45] D. Kaplan, "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More."

[46] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "CrypTen: Secure Multi-Party Computation Meets Machine Learning," Sep. 2022.

[47] T. Kocmi and C. Federmann, "Large Language Models Are State-of-the-Art Evaluators of Translation Quality," May 2023.

[48] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 611–626.

[49] Y. Lao, W. Zhao, P. Yang, and P. Li, "DeepAuth: A DNN Authentication Framework by Model-Unique and Fragile Signature Embedding," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, pp. 9595–9603, Jun. 2022.

[50] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proceedings of the Fifteenth European Conference on Computer Systems*, ser. EuroSys '20. New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 1–16.

[51] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, "Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network," *IEEE Access*, vol. 10, pp. 30 039–30 054, 2022.

[52] B. Li, V. Gadepally, S. Samsi, and D. Tiwari, "Characterizing Multi-Instance GPU for Machine Learning Workloads," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2022, pp. 724–731.

[53] X. Li, X. Li, C. Dall, R. Gu, J. Nieh, Y. Sait, and G. Stockwell, "Design and Verification of the Arm Confidential Compute Architecture," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, 2022, pp. 465–484.

[54] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '13. New York, NY, USA: Association for Computing Machinery, Jun. 2013, p. 1.

[55] M. Misono, D. Stavrakakis, N. Santos, and P. Bhatotia, "Confidential VMs Explained: An Empirical Analysis of AMD SEV-SNP and Intel TDX," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 3, pp. 36:1–36:42, Dec. 2024.

[56] S. Miwa and S. Matsuo, "Analyzing the Performance Impact of HPC Workloads with Gramine+SGX on 3rd Generation Xeon Scalable Processors," in *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, Nov. 2023, pp. 1850–1858.

[57] F. Mo, Z. Tarkhani, and H. Haddadi, "Machine Learning with Confidential Computing: A Systematization of Knowledge," Apr. 2023.

[58] 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 *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*. IEEE Computer Society, Jul. 2024, pp. 164–175.

[59] D. P. Mulligan, G. Petri, N. Spinale, G. Stockwell, and H. J. M. Vincent, "Confidential Computing—a brave new world," in *2021 International Symposium on Secure and Private Execution Environment Design (SEED)*, Sep. 2021, pp. 132–138.

[60] D. L. Mulnix, "Intel® Xeon® Processor Scalable Family Technical Overview," https://www.intel.com/content/www/us/en/developer/articles/technical/xeon-processor-scalable-family-technical-overview.html.

[61] S. Na, G. Jeong, B. H. Ahn, J. Young, T. Krishna, and H. Kim, "Understanding Performance Implications of LLM Inference on CPUs," in *2024 IEEE International Symposium on Workload Characterization (IISWC)*, Sep. 2024, pp. 169–180.

[62] R. Nertney, "Confidential Compute on NVIDIA Hopper H100 - Whitepaper," Tech. Rep., Jul. 2023.

[63] T. Ng, "Adobe Says It Won't Train AI Using Artists' Work. Creatives Aren't Convinced," *Wired*.

[64] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Ł. Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Ł. Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. d. A. B. Peres, M. Petrov, H. P. d. O. Pinto, Michael, Pokorny, M. Pokrass, V. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. J. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph, "GPT-4 Technical Report," Dec. 2023.

[65] A. Panwar, A. Prasad, and K. Gopinath, "Making Huge Pages Actually Useful," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '18. New York, NY, USA: Association for Computing Machinery, Mar. 2018, pp. 679–692.

[66] V. Patil, P. Hase, and M. Bansal, "Can Sensitive Information Be Deleted From LLMs? Objectives for Defending Against Extraction Attacks," in *The Twelfth International Conference on Learning Representations*, Oct. 2023.

[67] S. Pinto and N. Santos, "Demystifying Arm TrustZone: A Comprehensive Survey," *ACM Computing Surveys*, vol. 51, no. 6, pp. 130:1–130:36, Jan. 2019.

[68] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently Scaling Transformer Inference," *Proceedings of Machine Learning and Systems*, vol. 5, pp. 606–624, Mar. 2023.

[69] K. Rayner, E. R. Schotter, M. E. J. Masson, M. C. Potter, and R. Treiman, "So Much to Read, So Little Time: How Do We Read, and Can Speed Reading Help?" *Psychological Science in the Public Interest*, vol. 17, no. 1, pp. 4–34, May 2016.

[70] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Aug. 2015, pp. 57–64.

[71] M. Sallam, "ChatGPT Utility in Healthcare Education, Research, and Practice: Systematic Review on the Promising Perspectives and Valid Concerns," *Healthcare*, vol. 11, no. 6, p. 887, Jan. 2023.

[72] C. Segarra, T. Feldman-Fitzthum, D. Buono, and P. Pietzuch, "Serverless Confidential Containers: Challenges and Opportunities," in *Proceedings of the 2nd Workshop on SErverless Systems, Applications and MEthodologies*, ser. SESAME '24. New York, NY, USA: Association for Computing Machinery, Apr. 2024, pp. 32–40.

[73] O. Sharir, B. Peleg, and Y. Shoham, "The Cost of Training NLP Models: A Concise Overview," Apr. 2020.

[74] Y. Shen, H. Tian, Y. Chen, K. Chen, R. Wang, Y. Xu, Y. Xia, and S. Yan, "Occlum: Secure and Efficient Multitasking Inside a Single Enclave of Intel SGX," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20. New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 955–970.

[75] S. Szyller and N. Asokan, "Conflicting interactions among protection mechanisms for machine learning models," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'23/IAAI'23/EAAI'23, vol. 37. AAAI Press, Feb. 2023, pp. 15 179–15 187.

[76] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "DAWN: Dynamic Adversarial Watermarking of Neural Networks," in *Proceedings of the 29th ACM International Conference on Multimedia*, ser. MM '21. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 4417–4425.

[77] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models," in *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, Aug. 2021.

[78] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and Efficient Foundation Language Models," Feb. 2023.

[79] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open Foundation and Fine-Tuned Chat Models," Jul. 2023.

[80] F. Tramèr and D. Boneh, "Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware," Feb. 2019.

[81] C.-C. Tsai, D. E. Porter, and M. Vij, "{Graphene-SGX}: A Practical Library {OS} for Unmodified Applications on {SGX}," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 645–658.

[82] A. Viand and H. Shafagh, "Marble: Making Fully Homomorphic Encryption Accessible to All," in *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, ser. WAHC '18. New York, NY, USA: Association for Computing Machinery, Jan. 2018, pp. 49–60.

[83] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical*

Methods in Natural Language Processing: System Demonstrations. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[84] A. Wood, K. Najarian, and D. Kahrobaei, “Homomorphic Encryption for Machine Learning in Medicine and Bioinformatics,” *ACM Computing Surveys*, vol. 53, no. 4, pp. 70:1–70:35, Aug. 2020.

[85] S. Wu, H. Fei, L. Qu, W. Ji, and T.-S. Chua, “NExT-GPT: Any-to-Any Multimodal LLM,” Sep. 2023.

[86] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, “BloombergGPT: A Large Language Model for Finance,” Dec. 2023.

[87] M. Xue, Z. Wu, C. He, J. Wang, and W. Liu, “Active DNN IP Protection: A Novel User Fingerprint Management and DNN Authorization Control Technique,” in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Dec. 2020, pp. 975–982.

[88] M. Xue, Y. Zhang, J. Wang, and W. Liu, “Intellectual Property Protection for Deep Learning Models: Taxonomy, Methods, Attacks, and Evaluations,” *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 06, pp. 908–923, Dec. 2022.

[89] Y. Yang, M. Sonji, and A. Jog, “Dissecting Performance Overheads of Confidential Computing on GPU-based Systems,” in *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, May 2025, pp. 1–16.

[90] C. Zhao, D. Saifuding, H. Tian, Y. Zhang, and C. Xing, “On the Performance of Intel SGX,” in *2016 13th Web Information Systems and Applications Conference (WISA)*, Sep. 2016, pp. 184–187.

[91] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, “A Survey of Large Language Models,” Nov. 2023.