ConfBench: A Tool for Easy Evaluation of Confidential Virtual Machines

Andrea De Murtas [*†], Daniele Cono D'Elia [*], Giuseppe Antonio Di Luna [*]
Pascal Felber [†], Leonardo Querzoni [*], Valerio Schiavoni [†]
[*]Sapienza University of Rome, Rome, Italy, {delia,diluna,querzoni}@diag.uniroma1.it
[†]University of Neuchâtel, Neuchâtel, Switzerland, first.last@unine.ch

***Abstract*—Ensuring the security and confidentiality of cloud computing workloads is essential. To this end, major cloud providers offer computing instances based on trusted execution environments (TEEs) to support confidential computing in virtual machines. TEEs are hardware-based shielded environments building on technologies available today, such as Intel TDX or AMD SEV-SNP or that will soon be, as with ARM CCA.**

To lower the barriers to experimenting with these technologies for researchers and practitioners, we developed ConfBench, a tool for easy evaluation of confidential virtual machines. ConfBench supports both cloud-native workloads (Function-as-a-Service) and classic applications. ConfBench facilitates the management of the full lifecycle of such workloads, from their deployment to the gathering of performance metrics, taking into account the specifics of TEE-enabled confidential virtual machines. We use ConfBench to collect execution overhead measurements for different VM-enabled TEEs (Intel TDX and AMD SEV-SNP) through extensive experiments. We also showcase how ConfBench's architecture allows for validating also simulation-based TEEs, reporting preliminary results with ARM CCA. We highlight the intrinsic overheads of such confidential VMs by conducting stress tests against machine learning inference tasks, DBMS and native-OS operations benchmarking, as well as by evaluating the costs of attestation operations required in the context of confidential computing. The results indicate generally tenable overheads with modern TEEs, with exceptions mainly from I/O-intensive tasks, especially with TDX. ConfBench's multi-language support for FaaS workloads also lets us gain insights into differences stemming from varying complexities behind language runtimes. We release ConfBench to the research community and provide instructions to reproduce our experiments.

***Index Terms*—TEE, TDX, SEV-SNP, CCA, benchmarking, confidential computing**

I. Introduction

Cloud computing has revolutionized the way organizations deploy and manage applications, offering unparalleled scalability, flexibility, and cost-efficiency. However, one significant limitation persists: the inability of cloud providers to offer practical and widely accessible solutions for confidential computing. Existing services aimed at protecting data in use are limited in functionality [6], prohibitively expensive [30], or require significant technical expertise to implement [26].

Trusted Execution Environments (TEEs) have emerged as a promising solution to address the challenges of confidential computing [47]. First-generation TEEs, such as Intel SGX, were primarily designed to protect single processes. These environments, while offering certain security guarantees, impose complex implementation requirements on developers, requiring deep modifications to existing applications and a steep learning curve to ensure proper utilization. More recently, Intel's Trust Domain Extensions (TDX) [33], AMD's Secure Encrypted Virtualization (SEV-SNP) [7], and the announced ARM's Confidential Compute Architecture (CCA) [9] represent a significant shift towards the protection of entire virtualized subsystems, greatly simplifying the deployment of secure applications. The transition to these advanced TEEs lowers the barriers to entry for developers, doing away with intricate code modifications or error-prone annotations [58], [59]: programmers can now design standard applications, virtualize them, and deploy them within secure environments with minimal additional effort [19], [23], or even use confidential containers to further reduce deployment complexity [49].

The primary challenge this paper addresses is building ready-to-use tooling for assessing the performance and feasibility of running workloads in different (secure) architectures. To this end, we propose ConfBench, a system that can be used to conduct performance comparisons of different TEEs. ConfBench provides a framework for systematically testing and comparing the performance of user-supplied workloads in environments secured by TDX, SEV-SNP, and CCA.[1]

ConfBench supports both cloud-native workloads and more native and standard applications. For the former, one particularly promising application of these new TEEs is in the Function-as-a-Service (FaaS) computing paradigm [31]. FaaS has gained significant traction due to its ability to abstract infrastructure management and enable rapid scalability based on demand. Thanks to the stateless nature of FaaS workloads, cloud providers may opt to offer end-users a simpler and more effective option to achieve confidentiality of data in use: users could deploy their workloads in either a secure environment or a standard one depending on their specific security requirements, without the need for complex modifications to their codebase. As cloud providers consider integrating secure FaaS environments, understanding the implications on performance and operational efficiency of TEEs is crucial.

The key novelty of ConfBench is to facilitate performance evaluations within confidential VMs. It saves users from work

[1]ARM CCA is currently only available in a simulated environment as no supporting hardware was commercially available at the time of this writing. For the sake of this work, we rely on the reference FVP simulator [10].

in bringing up such VMs, masking differences with non-secure VMs and the TEE-specific provisions to enact.

To validate our proposed architecture, we develop a fully-functional prototype and use it to conduct experiments on cloud-native architectures for confidential virtual machines, putting on the bench TDX, SEV-SNP, and CCA. We benchmark a variety of both FaaS and traditional (*i.e.*, tasks involving DBMS, machine learning, and others) workloads implemented in various programming language and meant to reflect the diverse scenarios encountered in real-world applications. Through these comprehensive experiments, we gain insights into the performance characteristics and the feasibility of deploying these workloads in secure environments. The multi-language support, aligned with what FaaS providers usually offer to users, also allows us to appreciate differences ascribable to the different complexities of language runtimes.

Our experiments indicate that TDX is the most efficient technology overall, in particular for computational workloads. Compared to SEV-SNP, though, it exposes higher costs with I/O operations and attestation. The simulated CCA implementation instead consistently shows high overheads for every workload and may be viable only for relative comparisons within CCA confidential VMs. With FaaS workloads, the more complex language runtimes seem to impose a heavier burden on TEE operation.

In summary, this paper proposes two main contributions:

- CONFBENCH, a prototype system for easy execution of (FaaS) workloads in heterogeneous TEEs, which can be easily extended to support new TEEs and workloads.
- An evaluation of VM-based TEEs for the performance overhead from running tasks in confidential VMs offering new insights related to the nature of operations and, for FaaS workloads, the complexity of managed runtimes.

For experimental reproducibility, all our code, scripts, and datasets are available at https://doi.org/10.5281/zenodo.13349781.

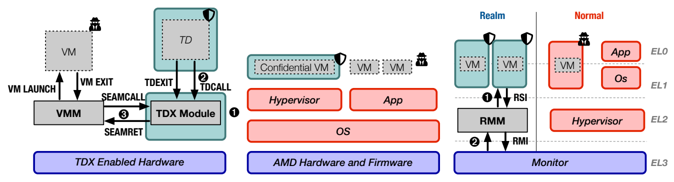
II. BACKGROUND

This section briefly introduces the main concepts behind TEEs and the three TEE variants considered in this study.

Trusted execution environments are isolated, tamper-resistant processing environments. They execute applications safely in a separation kernel [47], with enhanced security guarantees: the authenticity of the executed code, the integrity of the runtime states, and the confidentiality of the code.

Intel SGX [21] and AMD SEV [50] are widely adopted first-generation TEEs, enabling the instantiation of, respectively, enclaves and secure VMs. A new breed of TEEs (Intel TDX [17], AMD SEV-SNP [50], ARM CCA [12]) adds new security guarantees for fully confidential VMs, *e.g.*, memory encryption and integrity protection, and remote attestation.

An **attestation** primitive allows for establishing a trust relationship with a certain entity (*e.g.*, software or a system), by ensuring that it has not been tampered with, or that the genuine code is being executed. There are two types of attestation: local and remote. Local attestation proves integrity and authenticity



(a) Intel® TDX (b) AMD SEV-SNP (c) ARM® CCA

Fig. 1: TEEs considered in this work: TDX, SEV-SNP, CCA.

between co-located trusted environments, for example for establishing communication. Remote attestation involves three parties: the verifier, the attester, and a relying party. The verifier wants to verify the integrity of the attester. The relying party is the entity that will use the attester's services. The attester collects claims about its state, cryptographically signs them, and sends them to the verifier. The verifier then either accepts or rejects the attester based on the claims [41].

Intel TDX introduces hardware-level VM isolation (Trusted Domains), memory encryption, and remote attestation for VMs. Its core software component is the TDX Module (Fig. 1a–❶), which lives in a reserved memory space. The TDX Module runs in the Secure Arbitration Mode (SEAM) of the CPU [17] and it acts as an interface both for the hypervisor and for the Trusted Domains. The TDX Module always executes in SEAM root mode, which gives the Module privilege for interfacing with and managing TDs, while the TDs execute in SEAM non-root, but can switch to root mode using the assembly instruction `TDCALL` (Fig. 1a-❷). The hypervisor runs in VMX root mode. It can exchange with the TDX Module through a `SEAMCALL` instruction (Fig. 1a-❸), followed by a `SEAMRET` issued from the Module itself [17]. The memory of each TD is encrypted, integrity-checked, and isolated from every other untrusted components (*e.g.*, legacy VMs, other TDs, hypervisors), and can only be managed through the TDX Module. TDX supports the generation of a *quote* inside a TD with measurements showing the TD's state and the underlying infrastructure. A third party can use the quote to remotely attest system authenticity and integrity.

AMD SEV-SNP extends the VM memory encryption of AMD Secure Encrypted Virtualization (SEV) by including strong memory integrity protection, memory re-mapping, protection against side-channel attacks and attestation. These features improve both usability and protection [50] (Fig. 1b). Memory access control is enforced through the use of the Reverse Map Table (RMP), which keeps track of the owners of each page. If a VM wants to make data available to other entities, it can use shared unencrypted memory. Virtual Machine Privilege Levels (VMPLs) allow for having a guest VM's memory divided into four levels, ordered by the amount of privileges they possess, which is useful for nested virtualization [50]. Each SNP-enabled guest can request an attestation report to the underlying firmware, which is signed by the AMD-SP, a secure dedicated coprocessor [46], [50].

ARM CCA is a set of hardware and software features that enable the creation of confidential virtual machines (Fig. 1c).

In addition to TrustZone's normal and secure worlds [11], CCA introduces the *realm* and *root* worlds for enabling the instantiation of confidential VMs. Confidential VMs and the Realm Management Monitor (RMM) live in the realm world. These two entities operate on different exception levels (EL), sometimes referred to as privilege levels. CCA supports 4 physical addresses spaces, one for each security world. Address translation works in two stages: first, the OS translates a virtual address into an intermediate physical address, which in the second stage is translated to an actual physical address. The RMM manages stage-2 address translation; additionally, it exposes the Realm Services Interface (RSI) and the Realm Management Interface (RMI). Realms use the RSI (Fig. 1c-❶) to access services such as attestation and memory management. The host uses the RMI (Fig. 1c-❷) for managing confidential VMs [13]. For the remote attestation process, the realm gets an attestation report from the RMM, which contains relevant measurements (*e.g.*, the initial state of the realm and firmware components). This report can be cryptographically verified by the realm owner to trust the realm [13].

III. Tool Architecture

This section describes ConfBench, a tool for executing in a flexible manner FaaS and classic workloads through heterogeneous TEE architectures and languages. The architecture (Fig. 2) involves the following main components: a gateway, TEE-enabled hosts, confidential and non-confidential VMs.

A. Workflow and Main Components

Users can submit workloads to execute via a REST-based interface together with the corresponding runtime parameters (choice of a confidential VM or not, specific TEE, function instance parameters for FaaS workloads, *etc.*). ConfBench supports the three TEE platforms of §II as well as execution on non-confidential VMs to easily evaluate TEE overheads.

The **gateway** is the entry point for all requests. It receives and then dispatches workloads to appropriate execution platforms. Based on the query arguments, it selects a normal VM for execution or a secure VM with the desired TEE. ConfBench's design allows for easily adding new or additional execution platforms, following some constraints and file-system conventions (*e.g.*, directory structure, location of language interpreters and upload folders). The gateway executes the functions with the specified sets of arguments by invoking the appropriate commands on the remote hosts. Once completed, results are returned to the user.

The **hosts** are TEE-enabled hardware machines capable of instantiating confidential VMs. Hosts receive requests from the gateway, and, based on the query arguments (*i.e.*, destination port), they will route them to the appropriate destination.

We assume the gateway to have stateful knowledge about the hosts, their configuration, and appropriate access. Each host can run multiple types of VMs, either confidential or not.

To add a new TEE, one must provide all the relevant information needed by the gateway to access them. Internally,

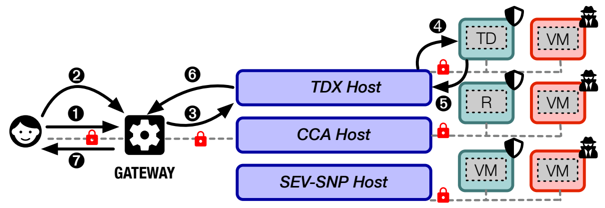


Fig. 2: ConfBench's architecture with sample run in TDX.

a dedicated gateway configuration file maps TEEs and their interface ports. The gateway maintains *TEE pools* to load-balance workload requests across different types of TEEs. Cloud provider users would adjust the load-balancing policy to their internal needs (*e.g.*, number of requests, SLAs).

In the FaaS scenario, the supported languages (listed in §IV-A) can be extended for a new one by preparing the relevant environment in the VMs, registering the new language in ConfBench, and implementing a **function launcher**. The latter must instantiate a runtime for the languages that need one, therefore its implementation is straightforward.

We highlight that in the case of non-FaaS scenarios, the user must cross-compile and submit the executable, unlike with FaaS where ConfBench streamlines the process.

B. Implementation Insights

Every VM on a host must have the same file locations, libraries, and interpreters to ensure consistent execution setups across VMs. In the ConfBench prototype, the gateway changes a network destination port, maintaining a mapping between ports and available VMs. The gateway component is implemented in Rust on top of the Axum [55] web framework. Each host machines relies on `socat` [28], a network relay tool to steer traffic to its hosted VMs. The implementation of ConfBench consists overall of 624 LOC in Rust.

While the setup of TDX and SEV-SNP confidential VMs is somehow streamlined [16] [8], the integration of ConfBench with simulated CCA realms required special attention.

At the time of our experiments, the FVP [10] simulator lacked proper documented support for networking (specifically, between the host and the VM). To overcome such limitations, we had to use a mix of `tap` and `tun` devices between the host and the simulator, and the simulator and the VMs.

For TDX, we initially observed consistently high overhead without a clear cause. Such issues were eventually solved by a firmware upgrade (`TDX_1.5.05.46.698`) published by Intel, boosting the execution runtime up to a $10\times$ factor.

The setup of the experimental machines was particularly time-consuming, due to the relatively recent availability of such technologies. We hope that our work can benefit other researchers looking for experimenting with these TEEs. Among the most significant technical challenges for the implementation, we stress how the specificities of each TEE requires ad-hoc mechanisms to plug into ConfBench's modular architecture, in particular due to the lack of common TEE standards (despite efforts such as the OpenEnclave initiative [4]) and to the different approaches to attestation [42].

ConfBench integrates with performance monitoring tools (Linux's `perf`) to measure and collect statistics of the various runs. Upon each function execution, it invokes `perf stat` when dispatching workloads. The collected performance metrics are piggybacked with the outputs returned to the users. The typical results, in addition to the wallclock time, include number of instructions executed, cache misses, *etc*. If performance counters are not available, such as inside CCA realms (hence `perf` cannot be used), one must rely on custom performance tools. ConfBench allows developers to extend the monitoring support with scripts, and we did so for CCA. Presently, ConfBench does not collect TEE-specific metrics (*e.g*., page faults), but adding scripts to track more metrics would be straightforward.

C. Example Run

The execution of a benchmark works as follows. Without loss of generality, we illustrate the scenario of a function execution inside a TDX trusted domain. First, the user uploads their function to the gateway (Fig.2-❶). The gateway maintains a database of available functions per supported language. Next, they send a request to indicate the target function, its arguments (Fig.2-❷), whether they want it to run in a confidential VM, and if so on what platform (TDX, SEV-SNP, CCA). The gateway sends the execution request to the chosen host (Fig.2-❸). The host receives the request and it routes it to the target VM (Fig.2-❹). The VM executes the function and sends the result back to the gateway (Fig.2-❺ and Fig.2-❻). The gateway forwards the results to the user (Fig.2-❼).

IV. Experimental Findings

In the following, we estimate the performance overheads of executing typical FaaS workloads and classic workloads in TEEs implemented by confidential VMs. For classic workloads, we use standard microbenchmarks, confidential machine learning inference benchmarks, and database stress tests.

A. Experimental Settings

We detail our experimental test bed for the TEEs next.

TDX: We use a server with an 8-core Intel Xeon Gold 5515+ CPU (3.20 GHz) and 64 GiB RAM. Trusted and legacy domains run Ubuntu 24.04 LTS (kernel 6.8.0-31-generic), while the TDX Module's version is `TDX_1.5.05.46.698`.

SEV-SNP: We use a server with a 16-core AMD EPYC 9124 CPU (3.0 GHz) and 64 GiB RAM. Secure and non-secure VMs run Ubuntu 22.04 LTS (kernel 6.5.0-41-generic).

CCA: Experimenting with CCA currently lacks real silicon, as confirmed also recently in online ARM events [24] and offline by ARM developers. As in recent related work [53], we rely on the official ARM Fixed Virtual Platform [10] simulator. ARM states in [14] that this simulator runs "at speeds comparable to the real hardware". Such a feat can enable researchers to achieve meaningful results, first and foremost for compatibility assessment (*i.e*., code can run correctly). Nevertheless, with simulation in general (beyond TEEs), we feel only relative comparisons within one simulator may be sound, and the real hardware should be used when available.

In each host we created two VMs: a VM with TEE-backed security guarantees and a "normal" VM. Both VMs enable a seamless execution of workloads, either as standalone binaries or via their corresponding language runtimes. For dispatching workloads to TEEs, the user interacts with the gateway, which runs on a separate machine (*i.e*., not any of the TEE ones).

B. Workloads

In this section, we describe the workloads used for the classic and FaaS benchmarks. Given the large diversity across the computing power of the hardware, unless otherwise stated, we systematically study the ratios between the confidential and the non-confidential execution time. To assess the differences across TEEs, we start with classic workloads deployed in a **confidential** setting, spanning heterogeneous tasks:

1) **ML**: we replicate the experiments and measurements from [51]. A pre-trained TensorFlow Lite model (MobileNetv1 [29]) is used to classify images. The dataset is from [51] and consists of 40 diversified 1-MB images
2) **DBMS:** We take a single-file version of the popular SQLite database, called *amalgamation* (*speedtest1.c*, v3460000), that includes its test suite to measure various aspects [52]. We compare the execution time across several tests, keeping the default relative test size (100).
3) **OS:** Byte UnixBench [2] consists of low-level system benchmarks for UNIX-like OSes. The suite attempts various tests and outputs a performance *index score* that compares the results with the suite's reference system.

Further, we study the performance of the **attestation** process for TDX and SEV which guarantees the integrity of the confidential code. As we discussed in Section II, the two follow fundamentally different approaches to it, hence we report the absolute latencies that users perceive from the completion of their steps. We used `go-tdx-guest` [27], a wrapper around the Intel library for attestation, and `spnguest` [56], a managing tool for SEV-SNP. We leave out CCA as the simulator lacks the required hardware support.

For the FaaS scenario, we recall such platforms allow end users to submit functions written in several different languages that are then interpreted or compiled. We consider the following languages and runtimes, which are a common choice in practice and industrial settings: **Python** (v3.12.3 for TDX, v3.10.12 for SEV-SNP, 3.11.8 for CCA), **Node.js** (v22.2.0 for TDX and SEV-SNP, v20.12.2 for CCA), **Ruby** (v3.2 for TDX, v3.0 for SEV-SNP, v3.3 for CCA), **Lua** (v5.4.6 for all) and **LuaJIT** (v2.1 for all), **Go** (v1.20.3 for all) and **Wasm** (Wasmi Engine v0.32 for all). We decided to evaluate these languages due to their practical relevance (aligning with what FaaS providers usually support) and to explore differences ascribable to managed runtimes. As shown later, lightweight runtimes (Lua) show lower overhead, while more complex ones (Python, JS) apparently impose a heavier burden on TEE operation (from memory integrity checking,

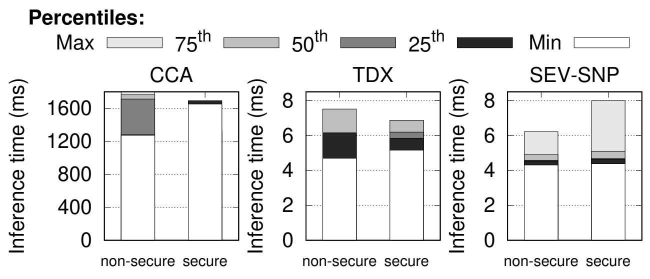


Fig. 3: Confidential ML workloads: distribution (as stacked percentiles) of the observed inference times.

encryption, *etc.*). This raises the question of whether future work could optimize runtime design, for example by placing parts of the managed runtime outside the TEE.

We consider a variety of open datasets for existing functions in these implementation languages. When not possible, we manually ported specific functions across languages, maintaining as much as possible the original logic. The execution mechanics of CONFBENCH, in particular via the function launchers (§III-A), allow for a common output across the diverse languages, easing the comparison efforts.

The chosen FaaS benchmarks rely on existing function benchmarks taken from FAASDOM suite [39], [40], FaaSbenchmark [1], Lua-Benchmarks [22], and wasmi-benchmarks [36]. Such benchmarks offer a large diversity of workload patterns (CPU, memory, I/O, *etc.*). We report results for 25 distinct workloads.

C. Classic Workloads

Confidential ML. We rely on the one of the default examples from the official TensorFlow GitHub repository [54]. This example deploys MobileNetV2, a small, low-latency model that can be used for image classification.[2] Fig. 3 presents, on a logarithmic scale, these results between the secure and non secure execution, for three different TEEs. We use a stacked percentile representation using tones of gray for the min, 25^{th}, median, 95^{th}, and max percentiles. Overheads for TDX and SEV-SNP are very similar, with TDX showing a limited advantage. For CPU-intensive tasks, TDX and SEV-SNP confidential VMs execute at close-to-native speed. Conversely, CCA introduces a larger overhead (up to 1.33× slower than a non-secure environment). In general, CPU-intensive tasks show less noticeable differences.

Confidential DBMS. We execute a large variety of tests from the official SQLite test suite [52]. The tests are composed of typical relational database operations, such as creating tables, inserting data, and querying it. We omit detailed plots for space and describe next the main findings. The overhead introduced by CCA is the largest ones, on average up to 10× compared to non-secure execution. Since this workload includes a large selection of heterogeneous task, this seems to impose a large strain on the CCA environment. Overheads for TDX and SEV-SNP are very similar and close to 1 (*i.e.*, execution times in secure and non-secure VMs are about the same). As the CCA tests run in a simulated environment, it is difficult to precisely debunk the root cause of such overheads. We suspect it is due to the two layers of abstraction used in our setup (as opposed to bare-metal execution in TDX and SEV), but it is hard to rule out other issues causing CCA to such negative results. Once real hardware becomes available, these tests shall be repeated.

[2]Although it targets mobile and embedded vision applications, we chose it as a well-known model of complexity comparable to the other benchmarks.

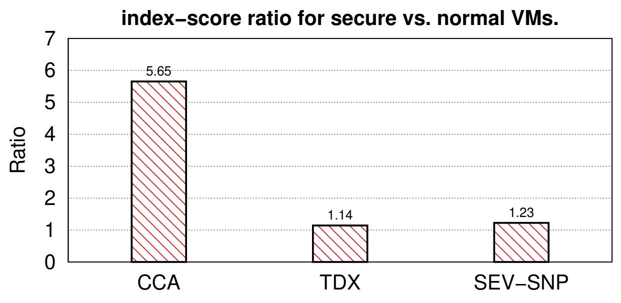


Fig. 4: UnixBench benchmarks.

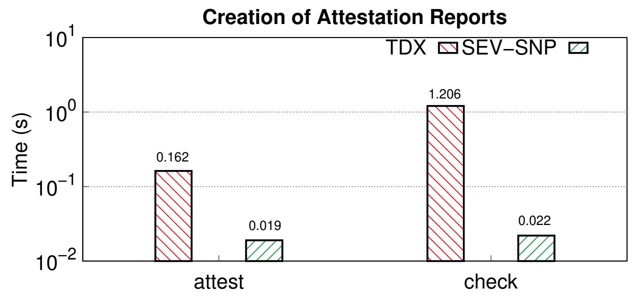


Fig. 5: Absolute times for the creation and validation of attestation reports in TDX and SEV-SNP. Y-axis uses a logarithmic scale.

Unixbench. This benchmark consists of a series of tests that measure the performance of the CPU, memory, disk, and file system. We run them in the single-threaded benchmark configuration. Each test returns an index score measuring performance compared to a baseline system (a SPARCstation 20-61 with 128 MB RAM, a SPARC Storage Array, and Solaris 2.3). We report the aggregated index reported by UnixBench out of the execution of several tests. These tests are very heterogeneous (*e.g.*, pipe-based context switching, file copy with different buffer size and maxblock, process creation *etc.*), thus giving us a good overview of the overall overhead at OS level when running in the TEEs. Fig. 4 compares execution times in secure and normal VMs, normalized as ratios. TDX introduces the least overhead, SEV-SNP leads to analogous figures, while CCA is the one introducing the most overhead. We note that the overheads with UnixBench are larger than in ML and DBMS workloads. Recent work [44] attributes slowdowns on UnixBench to frequent sleep and wake-up events that show in frequent TDVMCALL and VMEXIT events on, respectively, TDX and SEV-SNP confidential VMs.

Attestation. Fig. 5 reports the outcome of investigating

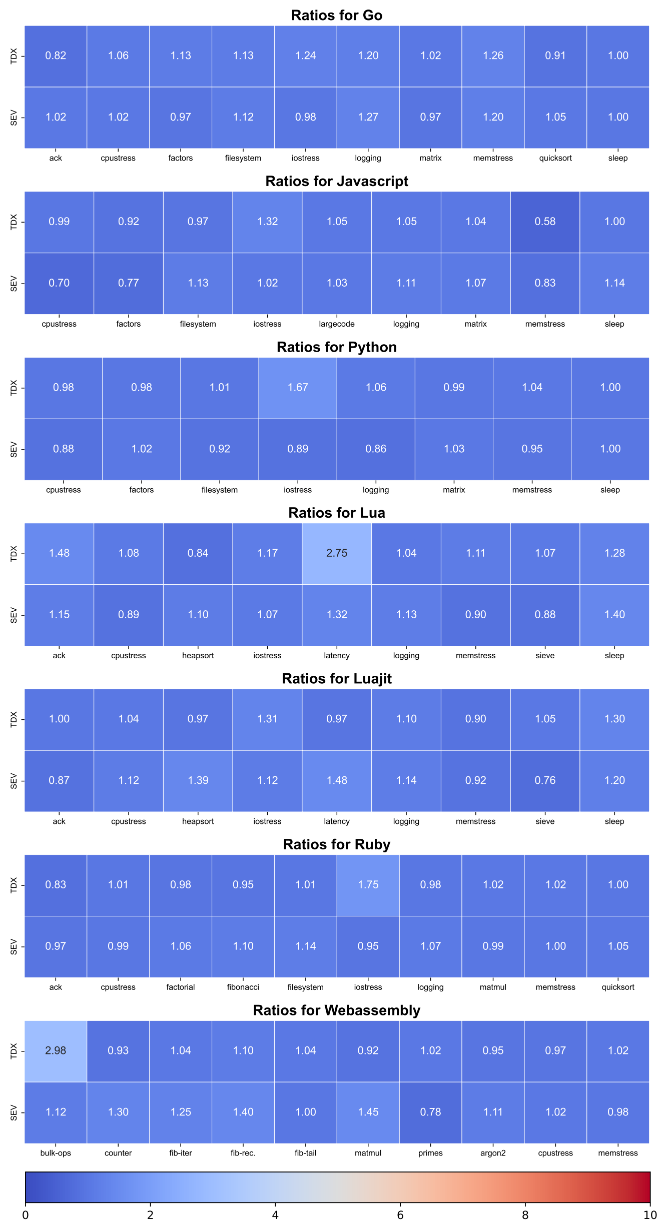


Fig. 6: TDX and SEV-SNP: ratios between mean execution times from secure and normal VMs for functions in different languages.'ack' stands for Ackermann function.

the costs of attestation, measured as wall-clock time. For TDX, we first setup the SGX DCAP (Data Center Attestation Primitives) [48], a set of libraries and tools for the creation and verification of attestation reports: we seek to generate a TD quote (§II). We show the costs of these operations in Fig. 5, "attest" step. The quote is then sent to a *verifier*. As mentioned, we use the `go-tdx-guest` wrapper [27] (further details in our repository). The verification process checks the root of trust of the platform and the correctness of the quote—cost shown in Fig. 5, "check" step.

For SEV-SNP, the process starts with the guest requesting an attestation report from the AMD Secure Processor firmware and follows a three-step process for its verification [46], [50].

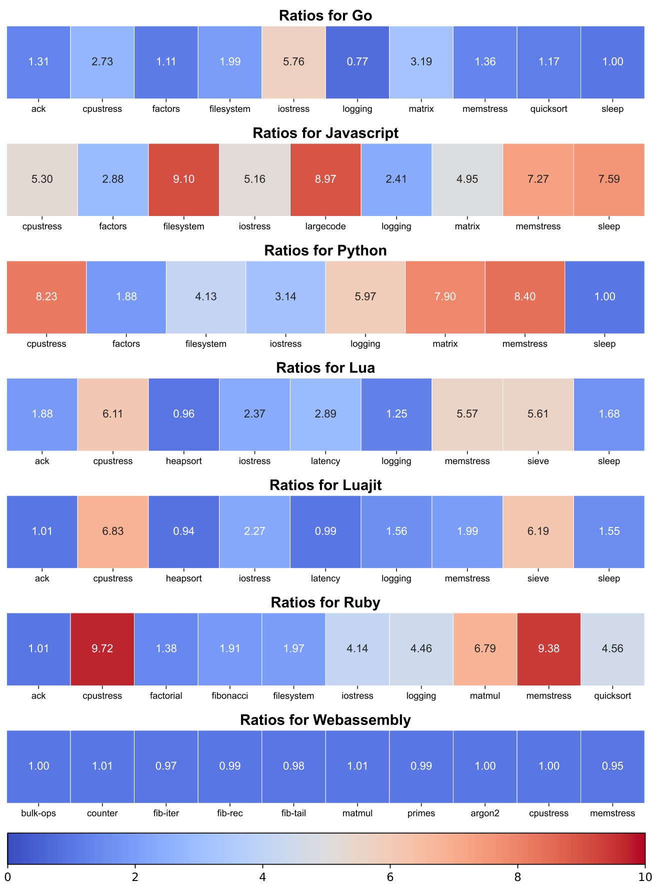


Fig. 7: CCA: ratios between mean execution times from secure and normal VMs for functions written in several languages.

Fig. 5 shows how both the generation ("attest") and validation ("check") phases are faster in SEV-SNP, which is expected due to the design differences. For the validation of the TDX quote, the implementation used retrieves TCB information and CRLs from the Intel PCS by making network requests [20], while the SEV-SNP implementation retrieves the required certificates from the underlying hardware.

D. *FaaS Workloads*

We collected performance figures when executing FaaS workloads using ConfBench. The functions deployed and executed in the TEEs are of different nature, and implemented in different implementation languages. Examples include:

- **cpustress**: intensive trigonometric calculations and arithmetic operations within a large iteration loop;
- **memstress**: repeated allocation of a 1-MB buffer so as to cover half of the machine's available memory;
- **iostress**: intensive read/write operations by creating and writing large files (1 MB) using the `dd` command[3];
- **logging**: print a large number of messages (3000);
- **factors**: compute the factors of a number;

[3]For the curious reader, we did not test with other sizes because I/O-bound workloads notoriously hit major bottlenecks [45]. Researchers have lately proposed improved I/O subsystems to overcome such limitations, shadowing more in-depth I/O benchmarking for the current design.

- **filesystem**: create and manage folders and files, performing read/write operations and cleanup. The workload creates two nested folders, then creates a file of 1 MB in the innermost one, writes to it, reads from it, and eventually deletes it, along with the folders.

We arrange this section in two parts: one for TDX/SEV-SNP followed by one for CCA. Preliminary measurements show how CCA incurs larger overheads in certain workloads, and understanding these differences compared to the bare-metal TEEs is difficult due to the simulation layer (in spite of it being claimed as hardware speed-accurate). Nevertheless, our CCA results still provide a first baseline for performance overheads.

We execute each function on both secure and normal VMs, using the same set of arguments. We measure the overhead introduced by running each function in a secure environment. We plot the ratio between the secure and non-secure counterpart. To minimize the impact of anomalies that might occur during a single run, we made 10 independent trials for each single function, and then used the average time results.

For every supported language, we prepared for CONFBENCH a workload-agnostic function launcher (§III-A) that reads the file where the function is defined and executes it using the given arguments. Our timing measurements exclude the time required by the launcher to bootstrap the runtime.

Specifically for Webassembly, we use the latest version of Wasmi, an efficient WASM interpreter written in Rust [37]. Most of its benchmark tests are taken from the benchmark suite used by Wasmi Labs themselves [36]. We extended this WASM benchmark suite with `cpustress` and `memstress`.

TDX and SEV-SNP. We use a heatmap representation to report our results, depicted in Fig. 6. Darker tones of blue indicate better ratios, computed for execution times in secure and normal VMs. The overhead introduced by the two TEEs is very similar: TDX tends to be faster with CPU and memory intensive workloads, while SEV-SNP is faster with I/O tasks. The specific connotation of intensive is given here by general traits of the code, with CPU-intensive being the most prevalent case. Our results are consistent with the baselines reported by Intel [34]. The overhead introduced by TDX for the `iostress` function is probably due to the use of encrypted bounce-buffers outside the Intel TDX protected memory space (*i.e.*, memory areas shared between TDs and the VMM). Bounce-buffers are known to introduce larger overheads when in use [34]. An upcoming TDX extension, TDX Connect [32], will improve functionality and performance of I/O virtualization: we expect these results to improve considerably. Finally, the attentive reader may notice how, in a few cases, the ratio is lower than 1, suggesting that execution in a secure VM is faster. We investigated some of these cases and observed higher amounts of cache line hits in runs in the secure VM. Modern TEE systems no longer face microarchitectural degradations typical of SGX. Recent studies such as TDXdown [57] explore induced variations in caching behavior, which may justify these apparently unexpected results.

CCA. We conclude our experiments by showing results from executing the same set of functions in the CCA environ-

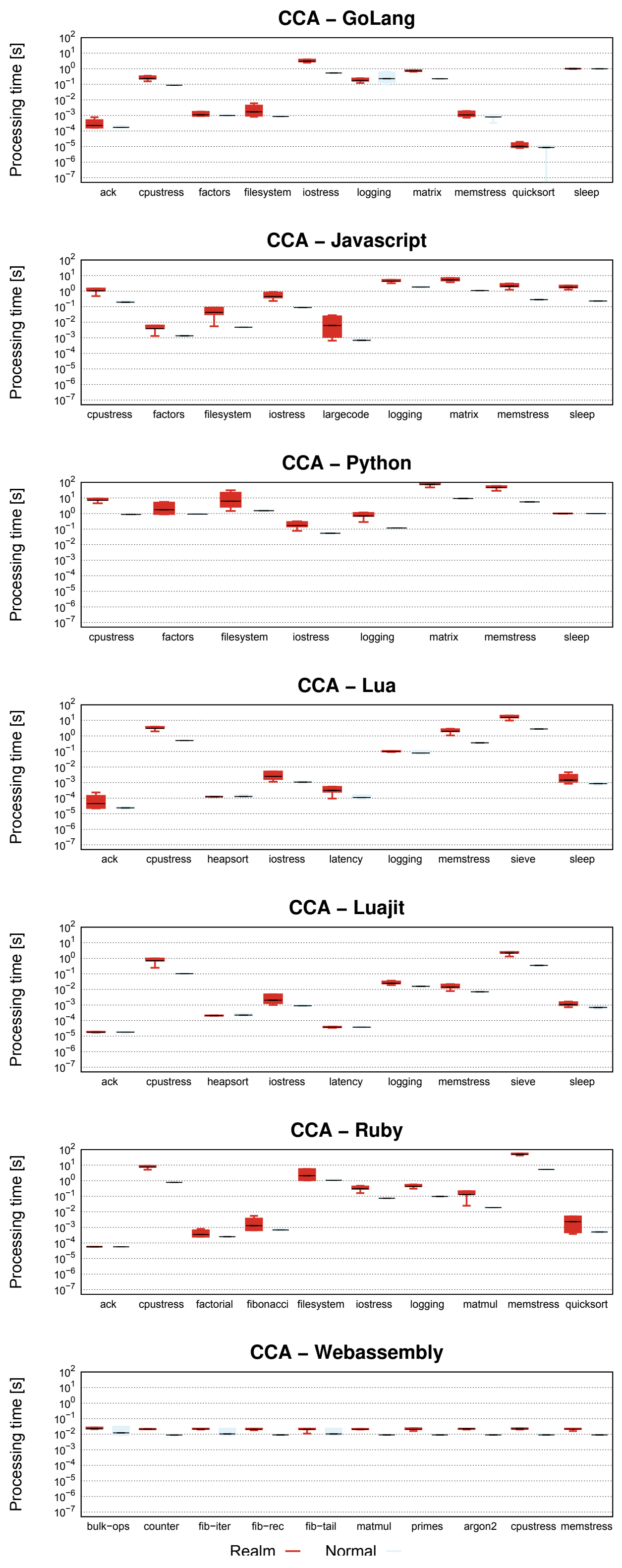


Fig. 8: CCA: distribution of execution times from secure and normal VMs for functions written in several languages.

ment. The software setup is the same as for the hardware-based TEEs. Inside the FVP simulator, we deploy two VMs (a secure realm and a non-secure one), thus introducing an additional layer of abstraction. We provide a heat-map in Fig. 7 built as with the other TEEs. Additionally, we provide a box-and-whiskers representation in Fig. 8 to report percentiles from all the 10 independent runs of each function. With confidential VMs, the length of the whiskers tends to be larger, indicating an increased variability in execution time. This variability was present also in the TDX and SEV-SNP experiments (plot omitted for space), but to a lesser extent.

Overall, CCA incurs much higher overheads compared to the other TEEs (visually, we see more lighter blue/red-ish cells). We find this can be attributed to the ARM machine being simulated. Similarly, a certain degree of inaccuracy of the time measurement introduced by the simulation itself might affect the reported ratios, and we plan to investigate such matters in future work. We choose to plot detailed data (Fig. 8) as, to the best of our knowledge, these experiments provide a first baseline for ARM CCA in literature. Once the hardware for CCA will become available on the market, it will be possible to revisit them and use ConfBench to determine the extent to which these overheads stem from the simulation versus the actual operation of CCA environments.

E. Discussion

Our experimental analysis highlights that code running in TEEs may often incur a minimal overhead. This proves the maturity of such technology and the degree of optimizations implemented inside these second-generation TEE architectures. TDX tends to be faster with CPU and memory intensive workloads, while SEV-SNP tends to be faster with I/O tasks, which may be due to the use of encrypted bounce-buffers outside the TDX protected memory space [3]. Tests on emulated CCA show unreliable performance, preventing us from drawing definitive conclusions on this matter.

V. Related Work

Cloud providers offer nowadays TEE-enabled VMs in their commercial offerings [15], [25], [43]. The authors of [60] propose an approach for reusable enclaves in the cloud to avoid cold start penalties, using Intel SGX, OpenWhisk, and WebAssembly. We note that integrating ConfBench with OpenWhisk or other FaaS frameworks remains a considerable engineering effort, which will become lower especially once real CCA hardware enters the market.

Intel SGX and OpenWhisk have been combined to realize a function-as-a-service framework in [5] with the security guarantees of TEEs. Similar to ConfBench, the framework allows measuring the performance of computing time inside SGX enclaves. ConfBench provides insights on the execution of functions inside a diverse set of next-generation confidential VMs, in addition to support for classic workloads.

Recent studies [18], [38] show how to deploy serverless workloads in SGX enclaves. However, to the best of our knowledge, we are the first to provide a full-fledge tool that both enables the execution of cloud-native workloads on next-generation confidential VMs, facilitating the evaluation of the performance of these workloads on different TEEs.

Serverless workloads can be deployed in confidential containers [49], however with unpractical results from the resulting overheads. Similar results can easily be reproduced leveraging ConfBench: we remark that its design can accomodate new types of confidential virtual machines, including containers and other types of execution environments.

VI. Conclusion and Future Work

This paper presented ConfBench, a tool that can be used to conduct smooth performance comparisons of different TEEs across a variety of workloads in a simple manner.

We showcased how to use it to easily deploy and run classical workloads and cloud-native ones. The architecture of ConfBench can easily be extended for current and future TEEs. We demonstrated the soundness of our design by running experiments on existing and future TEEs, including Intel TDX, AMD SEV-SNP, and (simulated) ARM CCA.

We provided experimental baselines using different types of micro- and macro-benchmarks, including confidential ML, confidential DBMS, and stress tests. Our results show that for more mature TEEs (*i.e.*, TDX and SEV-SNP), the overhead introduced by running benchmarks in confidential VMs is minimal. Perhaps counterintuitively, some scenarios achieve slightly better results inside confidential VMs rather than outside, an effect we traced back to differences in cache hits.

The performance achieved by executing in CCA realms with the reference FVP simulator appears poor if compared against non-secure virtual machines. We speculate that the root causes of such performance are to be found in the simulated nature of this environment. We will validate these simulated results once CCA hardware arrives to the consumer market.

We plan to extend this work along the following directions. Firstly, we intend to study the overheads of co-locating and executing several TEE-aware VMs inside the same host, as it happens in a typical cloud-based multi-tenant scenario. Secondly, we intend to provide researchers and practitioners support to easily deploy in ConfBench different types of confidential execution *units*, supporting native processes (for Intel SGX enclaves) or secure containers.

Given substantial engineering efforts, ConfBench would include native support for pluggable TEEs, integration to existing TEE monitoring libraries [35], and native integration with full-fledged FaaS platforms.

Acknowledgement

This work has been partially supported by projects SERICS (PE00000014) and Rome Technopole (ECS00000024) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

References

[1] FaaSBenchmark. https://github.com/nuweba/faasbenchmark. Accessed: 2025-04-17.

[2] UnixBench. https://github.com/kdlucas/byte-unixbench. Accessed: 2025-04-17.

[3] Performance Considerations of Intel® Trust Domain Extensions on 4th Generation Intel® Xeon® Scalable Processors. https://www.intel.com/content/www/us/en/developer/articles/technical/trust-domain-extensions-on-4th-gen-xeon-processors.html, September 2023.

[4] Open Enclave SDK. https://openenclave.io/sdk/, February 2025. Accessed: 2025-04-17.

[5] Fritz Alder, N. Asokan, Arseny Kurnikov, Andrew Paverd, and Michael Steiner. S-FaaS: Trustworthy and Accountable Function-as-a-Service using Intel SGX. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, CCSW'19, pages 185–199, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Amazon. AWS Nitro Enclaves. https://aws.amazon.com/ec2/nitro/nitro-enclaves/. Accessed: 2025-04-17.

[7] AMD. AMD Secure Encrypted Virtualization-Secure Nested Paging (SEV-SNP). https://www.amd.com/en/developer/sev.html. Accessed: 2025-04-17.

[8] AMDESE. AMD Secure Encrypted Virtualization. https://github.com/AMDESE/AMDSEV/tree/snp-latest. Accessed: 2025-04-17.

[9] ARM. ARM Confidential Compute Architecture. https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture. Accessed: 2025-04-17.

[10] ARM. Fast Models Fixed Virtual Platforms (FVP) Reference Guide. https://developer.arm.com/documentation/100966/1126/Introduction-to-FVPs?lang=en. Accessed: 2025-04-17.

[11] ARM. ARM TrustZone. https://documentation-service.arm.com/static/5f212796500e883ab8e74531, 2009. Accessed: 2025-04-17.

[12] ARM. ARM Realm Management Extension (RME) System Architecture. https://developer.arm.com/documentation/den0126/0100, 2021. Accessed: 2025-04-17.

[13] ARM. ARM Confidential Compute Architecture software stack. https://developer.arm.com/documentation/den0127/0200, 2023. Accessed: 2025-04-17.

[14] ARM Developers. Tools and Software: Fixed Virtual Platforms. https://developer.arm.com/Tools%20and%20Software/Fixed%20Virtual%20Platforms. Accessed: 2025-04-17.

[15] AWS. AWS Confidential Computing. https://aws.amazon.com/confidential-computing/. Accessed: 2025-04-17.

[16] Canonical. Intel Trust Domain Extensions (TDX) on Ubuntu 24.04. https://github.com/canonical/tdx. Accessed: 2025-04-17.

[17] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. Intel TDX demystified: A top-down approach. *ACM Computing Surveys*, 56(9):1–33, 2024.

[18] James Choncholas, Ketan Bhardwaj, and Ada Gavrilovska. TGh: A TEE/GC Hybrid Enabling Confidential FaaS Platforms. *arXiv preprint arXiv:2309.07764*, 2023.

[19] Luigi Coppolino, Salvatore D'Antonio, Davide Iasio, Giovanni Mazzeo, and Luigi Romano. An Experimental Evaluation of TEE Technology Evolution: Benchmarking Transparent Approaches based on SGX, SEV, and TDX. *arXiv preprint arXiv:2408.00443*, 2024.

[20] Intel Corporation. Intel Trust Domain Extensions. https://cdrdv2-public.intel.com/690419/TDX-Whitepaper-February2022.pdf, 2022. Accessed: 2025-04-17.

[21] V Costan. Intel SGX Explained. *IACR Cryptol, EPrint Arch*, 2016.

[22] Gabriel de Quadros Ligneul. Lua benchmarks. https://github.com/gligneul/Lua-Benchmarks. Accessed: 2025-04-17.

[23] Anna Galanou, Khushboo Bindlish, Luca Preibsch, Yvonne-Anne Pignolet, Christof Fetzer, and Rüdiger Kapitza. Trustworthy confidential virtual machines for the masses. In *Proceedings of the 24th International Middleware Conference*, Middleware '23, page 316–328, New York, NY, USA, 2023. Association for Computing Machinery.

[24] Paul Howard Gareht Stockwell, Nick Sample. Evolution of the ARM Confidential Compute Architecture. https://youtu.be/1AsvIt7bSLY?si=hwCVhyQs3I9XQ61V, March 2024.

[25] Google. Confidential VM overview. https://cloud.google.com/confidential-computing/confidential-vm/docs/confidential-vm-overview. Accessed: 2025-04-17.

[26] Google. Expanding our fully homomorphic encryption offering. https://developers.googleblog.com/en/expanding-our-fully-homomorphic-encryption-offering/. Accessed: 2025-04-17.

[27] Google. TDX Guest. https://github.com/google/go-tdx-guest. Accessed: 2025-04-17.

[28] Red Hat. Getting started with socat, a multipurpose relay tool for Linux. https://www.redhat.com/sysadmin/getting-started-socat. Accessed: 2025-04-17.

[29] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[30] IBM. Hyper protect crypto services. https://cloud.ibm.com/catalog/services/hyper-protect-crypto-services?catalog_query=aHR0cHM6Ly9jbG91ZC5pYm0uY29tL2NhdGFsb2cjc2VydmljZXM%3D. Accessed: 2025-04-17.

[31] IBM. What is function as a service (FaaS)? https://www.ibm.com/topics/faas. Accessed: 2025-04-17.

[32] Intel. Intel TDX connect architecture specification. https://www.intel.com/content/www/us/en/developer/articles/technical/trust-domain-extensions-on-4th-gen-xeon-processors.html. Accessed: 2025-04-17.

[33] Intel. Intel trust domain extensions (Intel TDX). https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html. Accessed: 2025-04-17.

[34] Intel. Performance Considerations of Intel Trust Domain Extensions on 4th Generation Intel Xeon Scalable Processors. https://www.intel.com/content/www/us/en/developer/articles/technical/trust-domain-extensions-on-4th-gen-xeon-processors.html. Accessed: 2025-04-17.

[35] Robert Krahn, Donald Dragoti, Franz Gregor, Do Le Quoc, Valerio Schiavoni, Pascal Felber, Clenimar Souza, Andrey Brito, and Christof Fetzer. TEEMon: A continuous performance monitoring framework for TEEs. In *Proceedings of the 21st International Middleware Conference*, Middleware '20, page 178–192, New York, NY, USA, 2020. Association for Computing Machinery.

[36] Wasmi Labs. Wasmi Benchmarking Suite. https://github.com/wasmi-labs/wasmi-benchmarks. Accessed: 2025-04-17.

[37] Wasmi Labs. Wasmi's new execution engine - faster than ever. https://wasmi-labs.github.io/blog/posts/wasmi-v0.32/. Accessed: 2025-04-17.

[38] Mingyu Li, Yubin Xia, and Haibo Chen. Confidential Serverless Made Efficient with Plug-In Enclaves. In *ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 306–318, 2021.

[39] Pascal Maissen. The FaaSdom benchmark suite. https://github.com/faas-benchmarking/faasdom. Accessed: 2025-04-17.

[40] Pascal Maissen, Pascal Felber, Peter Kropf, and Valerio Schiavoni. FaaSdom: a benchmark suite for serverless computing. In *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems*, DEBS '20, page 73–84, New York, NY, USA, 2020. Association for Computing Machinery.

[41] Jämes Ménétrey, Christian Göttel, Anum Khurshid, Marcelo Pasin, Pascal Felber, Valerio Schiavoni, and Shahid Raza. Attestation mechanisms for trusted execution environments demystified. In David Eyers and Spyros Voulgaris, editors, *Distributed Applications and Interoperable Systems*, pages 95–113, Cham, 2022. Springer International Publishing.

[42] Jämes Ménétrey, Christian Göttel, Anum Khurshid, Marcelo Pasin, Pascal Felber, Valerio Schiavoni, and Shahid Raza. Attestation mechanisms for trusted execution environments demystified. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 95–113. Springer, 2022.

[43] Microsoft. Azure Confidential VM.. https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview. Accessed: 2025-04-17.

[44] Masanori Misono, Dimitrios Stavrakakis, Nuno Santos, and Pramod Bhatotia. Confidential vms explained: An empirical analysis of amd sev-snp and intel tdx. *Proc. ACM Meas. Anal. Comput. Syst.*, 8(3), December 2024.

[45] Masanori Misono, Dimitrios Stavrakakis, Nuno Santos, and Pramod Bhatotia. Confidential VMs Explained: An Empirical Analysis of AMD SEV-SNP and Intel TDX. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 8(3):1–42, 2024.

[46] Petar Paradžik, Ante Derek, and Marko Horvat. Formal Security Analysis of the AMD SEV-SNP Software Interface, 2024.

[47] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64, 2015.

[48] Vinnie Scarlata, Simon Johnson, James Beaney, and Piotr Zmijewski. Supporting third party attestation for Intel SGX with Intel data center attestation primitives. *White paper*, 12, 2018.

[49] Carlos Segarra, Tobin Feldman-Fitzthum, Daniele Buono, and Peter Pietzuch. Serverless Confidential Containers: Challenges and Opportunities. In *Proceedings of the 2nd Workshop on SErverless Systems, Applications and MEthodologies*, SESAME '24, pages 32–40, New York, NY, USA, 2024. Association for Computing Machinery.

[50] Sev-SNP, AMD. Strengthening VM isolation with integrity protection and more. *White Paper, January*, 53:1450–1465, 2020.

[51] Sandra Siby, Sina Abdollahi, Mohammad Maheri, Marios Kogias, and Hamed Haddadi. GuaranTEE: Towards Attestable and Private ML with CCA. In *Proceedings of the 4th Workshop on Machine Learning and Systems*, EuroMLSys '24, page 1–9, New York, NY, USA, 2024. Association for Computing Machinery.

[52] SQLite. speedtest1.c. https://sqlite.org/src/file/test/speedtest1.c. Accessed: 2025-04-17.

[53] Supraja Sridhara, Andrin Bertschi, Benedict Schlüter, Mark Kuhne, Fabio Aliberti, and Shweta Shinde. ACAI: protecting accelerator execution with arm confidential computing architecture. In *Proceedings of the 33rd USENIX Conference on Security Symposium*, SEC '24, USA, 2024. USENIX Association.

[54] Tensorflow. Tensorflow lite c++ image classification demo. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/examples/label_image. Accessed: 2025-04-17.

[55] tokio rs. Axum. https://github.com/tokio-rs/axum. Accessed: 2025-04-17.

[56] VirTEE. snpguest. https://github.com/virtee/snpguest. Accessed: 2025-04-17.

[57] Luca Wilke, Florian Sieck, and Thomas Eisenbarth. TDXdown: Single-Stepping and Instruction Counting Attacks against Intel TDX. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 79–93, 2024.

[58] Peterson Yuhala, Pascal Felber, Hugo Guiroux, Jean-Pierre Lozi, Alain Tchana, Valerio Schiavoni, and Gaël Thomas. SecV: Secure Code Partitioning via Multi-Language Secure Values. In *Proceedings of the 24th International Middleware Conference*, Middleware '23, page 207–219, New York, NY, USA, 2023. Association for Computing Machinery.

[59] Peterson Yuhala, Jämes Ménétrey, Pascal Felber, Valerio Schiavoni, Alain Tchana, Gaël Thomas, Hugo Guiroux, and Jean-Pierre Lozi. Montsalvat: Intel SGX shielding for GraalVM native images. In *Proceedings of the 22nd International Middleware Conference*, Middleware '21, page 352–364. Association for Computing Machinery, 2021.

[60] Shixuan Zhao, Pinshen Xu, Guoxing Chen, Mengya Zhang, Yinqian Zhang, and Zhiqiang Lin. Reusable Enclaves for Confidential Serverless Computing. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4015–4032, Anaheim, CA, August 2023. USENIX Association.