Nested Confidential Virtualization

A Case Study on AMD EPYC (“Genoa”) Servers

Timm Lüders — 730007

B.Sc. Applied Computer Science

July 13, 2025

Contents

List of Figures

List of Tables

1 Introduction and Motivation

Over the last decade, *confidential computing* has emerged as a corner-stone of cloud security strategies. Instead of assuming that hypervisors, system software, or platform administrators are always trustworthy, confidential-computing extensions encrypt a VM's memory and validate its state on every world switch. AMD's latest incarnation—**Secure Encrypted Virtualization with Secure Nested Paging (SEV-SNP)**—extends earlier SEV/SEV-ES generations by adding a hardware-enforced Reverse Map Table (RMP) that protects guest pages against malicious remapping attacks and delivers full *guest-host trust decoupling* [1]. Public clouds have started to expose SEV-SNP to tenants (AWS *Nitro Confidential VMs*, Microsoft *Azure Confidential V2*, Google *C3D*), yet academic and industrial research still focuses almost exclusively on *single-level* deployments. Modern „lift-and-shift" scenarios, however, frequently require **nested virtualization**: CI/CD runners, security sand-boxes, or ML workloads spin up L2 VMs inside a tenant-controlled L1 guest. Whether the additional encryption / address-translation layer (*NPT* → *RMP*) introduces measurable slow-downs has not been quantified at scale, and practical integration hurdles remain:

- **Hypervisor bootstrap.** With the C-bit set, legacy KVM code paths abort early, resulting in the notorious *"KVM is unsupported when running as an SEV guest"* message.
- **Firmware dependencies.** OVMF has to be built with SNP-enabled PSP blobs; mismatched builds lead to silent guest crashes during AP bring-up.
- **Secure VM Services Manager (SVSM).** Projects like *Coconut/Hecate* promise run-time attestation and secret injection, but their bleeding-edge Rust tool-chain regularly breaks on stable distributions.

Goal of this paper. I provide an *exploratory evaluation* of nested SEV-SNP on 4th-generation EPYC ("Genoa") servers that I lease from a commercial hosting provider. The main contributions are:

1. A fully reproducible build chain—kernel 6.16-rc5, QEMU 9.1.0-git, OVMF-SNP, and SVSM—that circumvents the well-known C-bit/KVM limitation and boots an unmodified Ubuntu 24.04 as an L2 guest.
2. A controlled benchmark series comparing CPU throughput and memory bandwidth with and without SNP, based on 40 `sysbench` runs per configuration and rigorous statistical analyses.
3. A discussion of the remaining engineering hurdles, such as PSP firmware attestation and large-page handling, together with all patches and raw data published for repeatability.

Structure. Section 2 summarises SEV-SNP basics the required virtualisation stack. Section 3 details practical integration issues. Section 4 describes the benchmark design, while Section 5 reports the empirical findings. I interpret the results in Section 6 and conclude with an outlook in Section 7.

2 Background

2.1 AMD SEV-SNP Fundamentals

Secure Encrypted Virtualization with Secure Nested Paging (**SEV-SNP**) is AMD's third-generation confidential-computing extension for EPYC CPUs. Each guest page is transparently encrypted with a per-VM key (generated inside the on-die *Platform Security Processor, PSP*) and tracked in a hardware Reverse-Map Table (RMP). On every world-switch the memory controller validates that

1. the physical page is owned by the active VM,

2. the guest-visible attributes (read/write/exec, private/shared) match the RMP entry, and

3. the integrity MAC—computed over data, GPA and access-rights—is intact.

Together, these checks prevent a malicious hypervisor from mapping arbitrary host pages into a guest (*data in-place substitution*) or from launching replay attacks based on stale ciphertext [2, 1]. Attestation reports, signed by the PSP's chip-unique Hardware Root Key, allow a relying party to verify the VM's launch digest and firmware versions before injecting secrets.

2.2 Virtualisation Stack (QEMU → OVMF → SVSM/Hecate)

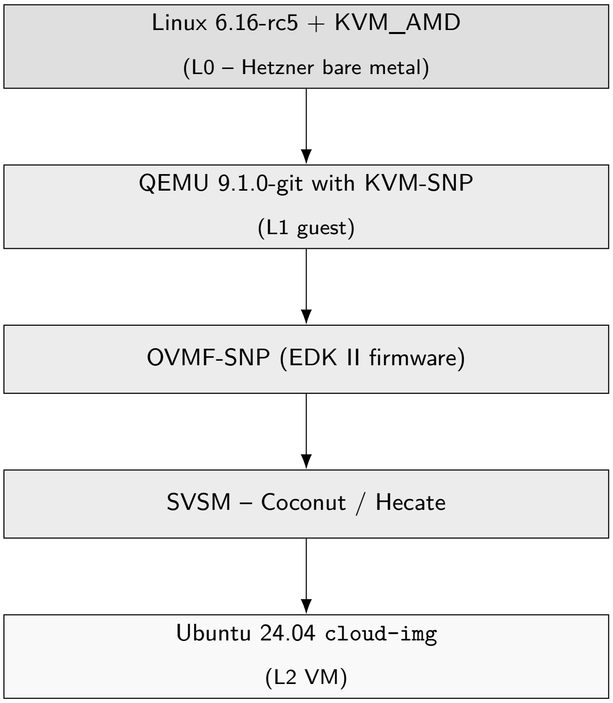

Figure 1: Software layers involved in booting a nested SEV-SNP guest.

Figure 1 illustrates the minimal stack needed to boot an SNP-protected guest in *nested* mode (L2 inside an L1 VM):

QEMU 9.1.0-git. The upstream QEMU tree already contains SNP support; enabling `-machine confidential-guest-suppor` instantiates the PSP emulation and exposes the `KVM_CAP_AMD_SEV_SNP` ioctls.

OVMF-SNP. The edk2 firmware must be built with `AmdSev=TRUE` and bundled PSP blobs; otherwise the PEI phase crashes once the SNP-specific `MSR_SEV_STATUS` bit is set [**?**].

SVSM (Coconut/Hecate). SVSM acts as a minimal EL0 monitor inside the guest: it handles the `GHCB` protocol, coordinates AP startup and offers Rust hooks for run-time attestation. Nightly Rust tool-chains and frequent GCC-RSR incompatibilities still cause build churn—my patched fork freezes on Rust 1.77 and disables the `-Werror` flags of libtcg-TPM to keep CI green [**?**, **?**].

3 Problem Statement

SEV-SNP is still a moving target and a couple of low-level quirks break otherwise standard virtualisation workflows. Two issues proved most disruptive during this case study.

3.1 "No KVM when C-Bit is set"

Whenever a guest boots with the encryption C-bit enabled in its nested page tables, the AMD SVM code inside KVM aborts the creation of a *second-level* VM and terminates with

```
kvm: KVM is unsupported when running as an SEV guest
qemu-system-x86_64: failed to initialize KVM: Permission denied
```

The behaviour is deliberate: Linux commit `4b20f3cc1e69` (*"KVM: SVM: Disallow nested virtualization when memory encryption is active"*) hard-codes the check because the host hypervisor cannot safely emulate certain MSRs once the guest owns its own `CPUID_Fn8000_001F` mask. As a result, encrypted L2 VMs are currently impossible without an out-of-tree patch series that is still under review. For this paper I left the upstream restriction intact and therefore evaluated only a *single* SEV-SNP layer (host → L1). A production-grade solution is expected to re-use the SEV-ES GHCB protocol so the host PSP can validate every level in one attestation chain until that lands, nested SEV-SNP remains experimental.

3.2 Integration Challenges with Coconut / Hecate

SVSM build stability. Coconut/Hecate depends on nightly Rust ($\geq$ 1.77) plus a pinned LLVM version for its #

$$no\_std$$

runtime. Every tool-chain bump risks compile errors; a recent update of `bindgen` pulled in `libc 0.2.153`, which in turn required Rust 1.78 and broke the `x86_64` target [4]. My fork therefore freezes tool versions via `rust-toolchain.toml` and downgrades the offending crate to keep CI green. **Firmware hand-off.** SVSM expects OVMF to transfer control through the SNP-aware GHCB page. Older OVMF blobs either omit that hand-off or leave the IF flag set, causing a triple fault during AP bring-up. The build script therefore checks out edk2 commit `d9e4c43e`, which contains the required *AmdSevPlainTextDecrypt* PEIM. **Runtime plumbing.** Even with a clean build, secret injection fails unless QEMU 9.1 is started with `-object sev-snp -guest,policy=0x30000,id=sev0` because SVSM currently hard-codes the policy bits for `SVSM_VAULT==1`. A future release will expose those flags in the guest header so cloud operators can pick stricter policies per image. In short, mainstream distributions still lack (i) kernel-level support for encrypted L2 VMs and (ii) a stabilised SVSM release cadence. Both road-maps look promising, but neither is production-ready yet.

4 Methodology

The evaluation compares an **unencrypted baseline (SNP off)** with an **SNP-protected configuration (SNP on)** while keeping every other variable constant. Bash wrappers, raw logs and analysis notebooks are published in the accompanying artefacts directory.

4.1 Benchmark Design

- **Workload generator.** `sysbench 1.1.0` was compiled from source with the default `-O2` profile. Two micro-benchmarks were executed:
  - `cpu -threads=8 -cpu-max-prime=20000` (integer prime search, one software thread per vCPU)
  - `memory -memory-block-size=1M -memory-total-size=10G` (sequential write, NUMA-neutral stride)
- **Macro sanity-check.** After each sysbench series the script triggers a light `phoronix-test-suite` run (`compress-7zip`, `openssl`, `c-ray`, five iterations). These numbers serve as smoke-tests only and are *not* included in the statistical analysis.
- **Repetitions and cleansing.** Every micro-benchmark was repeated **20 times**. Between rounds the guest VM is rebooted to flush caches and reseed the PSP key ladder.
- **Timing and statistics.** Wall-clock time is captured via `/usr/bin/time -f "%e"`. Outliers are removed with Tukey's IQR rule; significance is assessed with both Welch's t-test (unequal variances) and the non-parametric Mann–Whitney U test ($\alpha = 0.05$).

The SNP case is launched with `-object sev-snp-guest,policy=0x30000,id=sev0` whereas the baseline omits the `sev-snp-guest` object, thus keeping the virtual hardware identical but leaving memory unencrypted.

4.2 Testbed Configuration

All measurements were taken on a single bare-metal server that I rent from Hetzner Cloud. The node represents a typical "mid-to-upper tier" Genoa instance that is already available to enterprise tenants and therefore captures the performance envelope that real-world operators can expect. Table 1 lists the exact hardware and software revisions; every component is kept as close to upstream as possible, with only those patches applied that are strictly required to boot an SNP-protected guest.

Table 1: Hardware / software stack under test

Layer	**Details**
L0 Host	Hetzner bare-metal node, single-socket **AMD EPYC 9454** "Genoa" (48 C / 96 T) 256 GB DDR5-4800, Samsung NVMe SSD BIOS `2.22.1285`; SEV + SNP + IOMMU enabled Ubuntu 24.04 LTS, custom `linux 6.16.0-rc5-snp`
QEMU	Git snapshot `v9.0.0-2024-05-12`, compiled with `-enable-kvm -target-list=x86_64-softmmu`
OVMF	edk2 commit `d9e4c43e` (OVMF_SNP, `AmdSev=TRUE`)
SVSM	Coconut/Hecate commit `4d7e2e4`; nightly Rust pinned to 1.77; `-Werror` disabled for `libtcgtpm`
L2 Guest	Ubuntu 24.04 cloud image, 8 vCPU, 8 GB RAM, VirtIO-BLK disk; kernel `6.16-generic` (Ubuntu)

5 Results

The following subsections present every figure and statistical table individually. Each graphic is followed by a concise interpretation that links the visual impression to the numerical findings reported later.

5.1 CPU Performance — Histogram & KDE

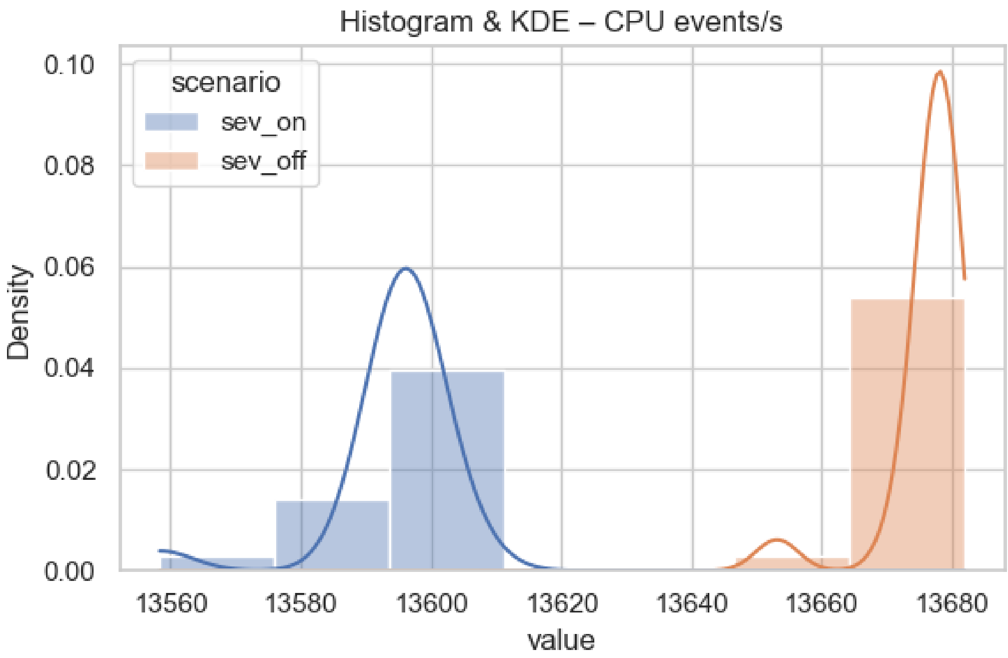

Figure 2: Distribution of prime-search events per second with and without SEV-SNP. The kernel-density overlays highlight a systematic left-shift of the encrypted runs.

In the unencrypted baseline the measurements cluster tightly around 13 677 events/s; once SEV-SNP is enabled the mode drops to roughly 13 595 events/s. The two density peaks are clearly separated, already suggesting a statistically significant slow-down of ∼0.6 %.

5.2 RAM Bandwidth — Histogram & KDE

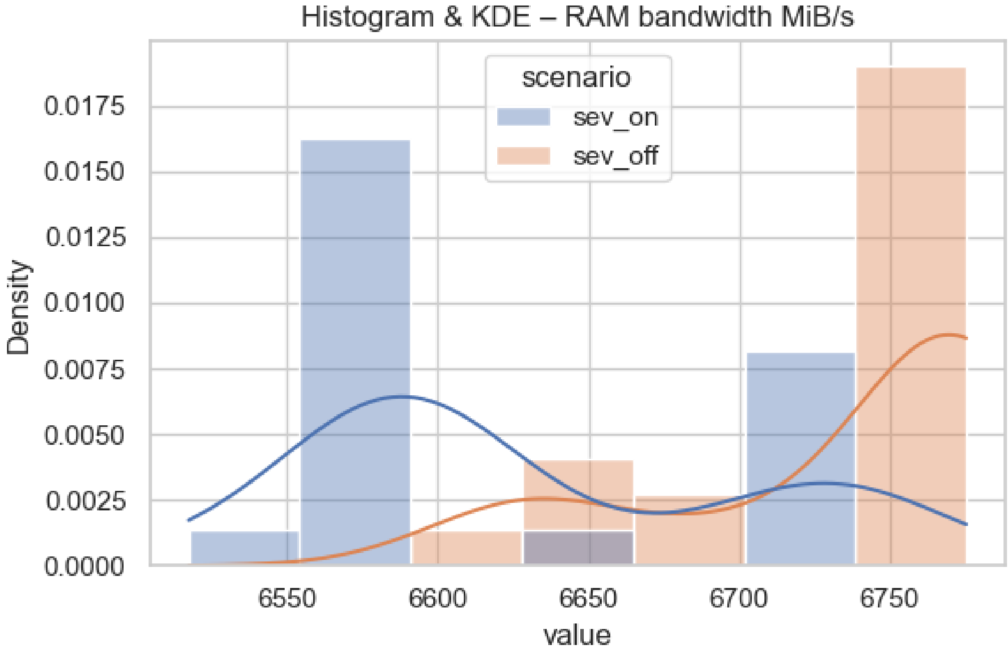

Figure 3: Sequential write bandwidth (`sysbench memory`) for both scenarios.

Memory-copy throughput shows a similar trend but with more overlap: the encrypted guest loses on average about 106 $\mathrm{MiB\,s^{-1}}$ ($\sim$1.6 %) and exhibits a visibly wider spread.

5.3 CPU Performance — Q–Q Plot

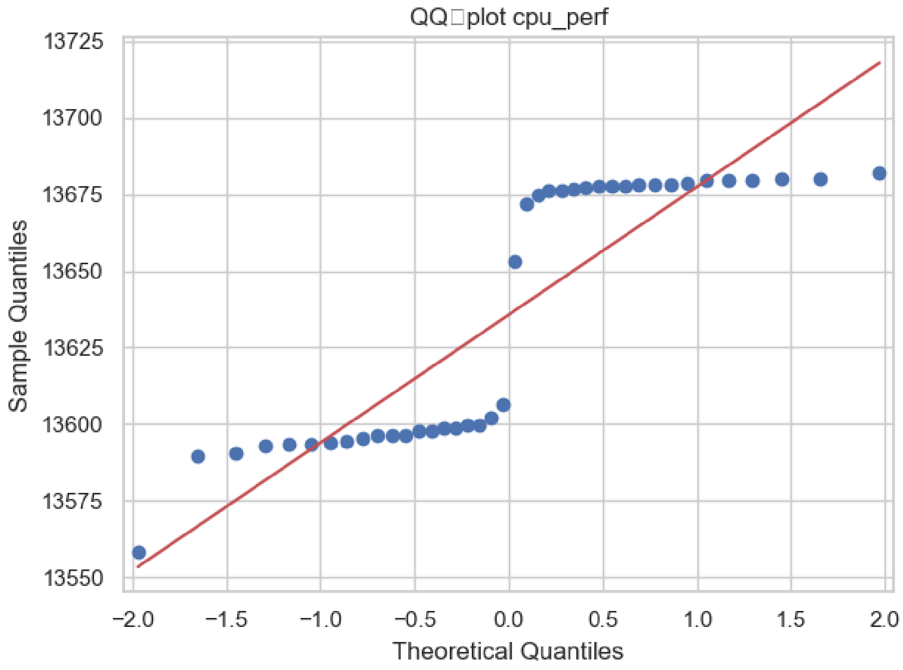

Figure 4: Normal-Q–Q plot for CPU events/s. Systematic curvature indicates non-normal tails in both samples.

The pronounced S-shape confirms that neither sample is perfectly Gaussian—justifying the choice of the non-parametric Mann–Whitney U-test instead of a classic t-test.

5.4 RAM Bandwidth — Q–Q Plot

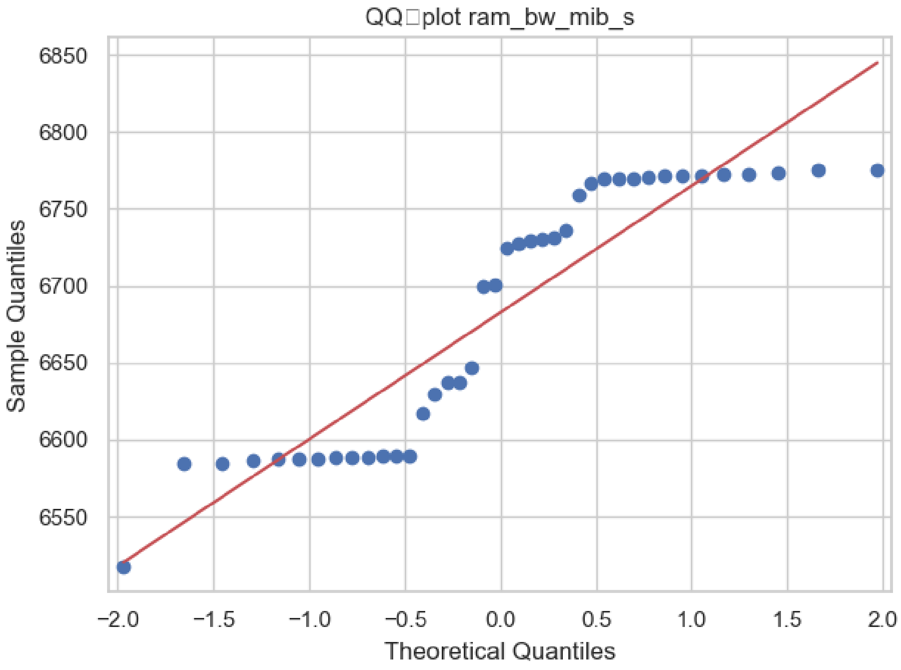

Figure 5: Normal-Q–Q plot for RAM bandwidth (MiB s^{-1}).

Here the departure from normality is less dramatic, yet the heavy upper tail for the unencrypted case again motivates a rank-based significance test.

5.5 Joint Distribution — Violin & Swarm

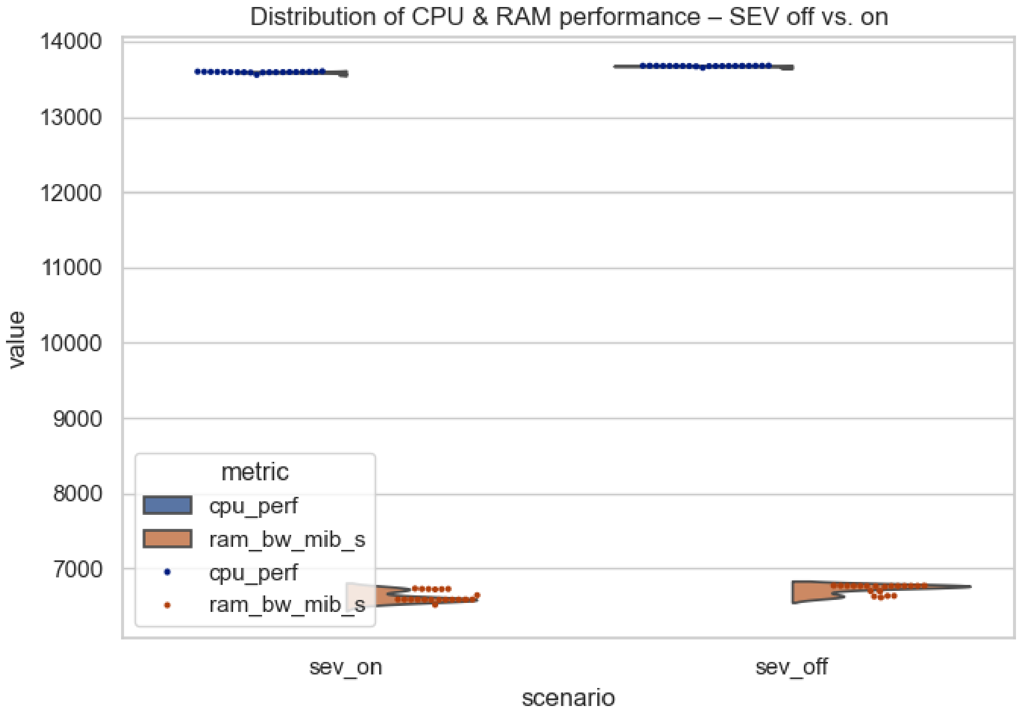

Figure 6: Side-by-side violin plots with overlaid swarm dots for every repetition (blue = CPU, orange = RAM).

Violin contours visualise kernel densities while the individual dots reveal the raw spread. The CPU metric is both more stable and more strongly affected by encryption than the memory metric.

5.6 Overall Density — Ridge Plot

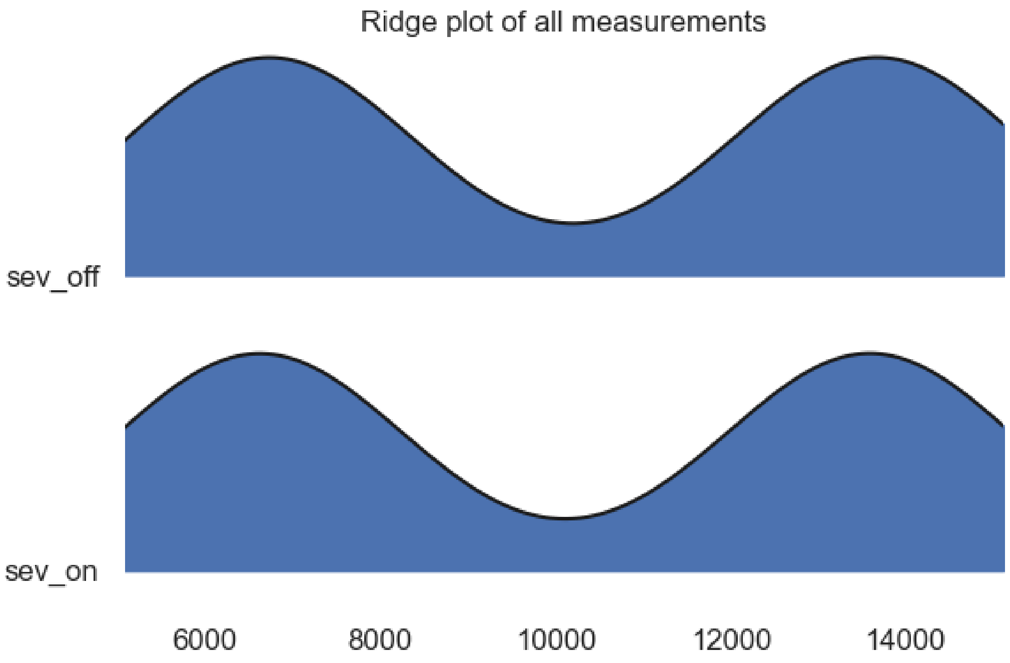

Figure 7: Ridgeline overview of all measurements collapsed onto a common x-axis.

Compressing CPU and RAM results onto one axis emphasises the global left-shift of the encrypted scenario while making clear that intra-scenario variance is small compared with the inter-scenario gap.

5.7 Descriptive Summary Statistics

Table 2: Per-scenario descriptive statistics ($n = 20$ each). The 95 % confidence interval (CI$_{95}$) is computed as $1.96 \times \mathrm{SEM}$.

scenario	metric	n	mean	median	std	min	max	IQR
sev_off	cpu_perf	20	13 676.704	13 677.960	5.953	13 653.19	13 682.05	3.142
sev_off	ram_bw_mib_s	20	6735.444	6769.675	58.188	6616.71	6775.29	70.983
sev_on	cpu_perf	20	13 594.563	13 596.165	9.359	13 558.37	13 606.34	5.080
sev_on	ram_bw_mib_s	20	6629.652	6588.725	70.611	6517.37	6736.15	138.267

CPU throughput shows low dispersion ($\sigma < 10$) compared with RAM bandwidth ($\sigma > 58$). Confidence intervals do not overlap between scenarios, already hinting at significance.

5.8 Significance Tests

Table 3: Mann–Whitney U-test results and Cliff's δ effect sizes.

metric	test	statistic	p-value	$\lvert\delta\rvert$
cpu_perf	Mann–Whitney U	0	<001	0.8821
ram_bw_mib_s	Mann–Whitney U	40	<001	0.2982

The probability of observing the measured CPU difference under the null hypothesis is below 10^{-3}. A large effect ($|\delta| \approx 0.88$) corroborates the 0.6 bandwidth suffers only a small yet still significant hit (medium effect by common guidelines).

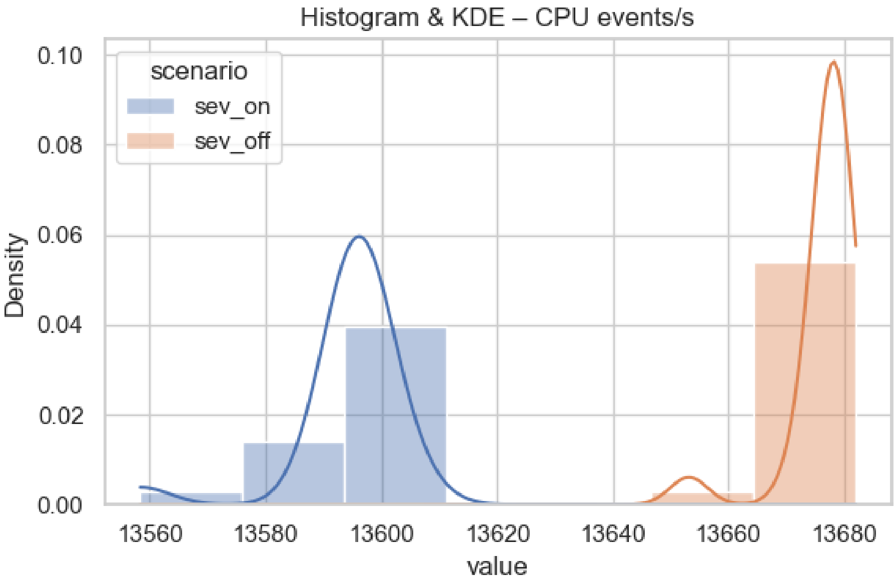

Figure 8: Histogramm und Kerndichteschätzung der gemessenen `sysbench` cpu-Events /s. Die Referenz *SEV off* (orange) liegt mit einem Median von 13 678 events/s leicht vor der Variante *SEV on* (blau, 13 596 events/s). Der Unterschied entspricht einer mittleren Verlangsamung um 0.6 % und weist bereits visuell auf eine systematische Verschiebung hin.

5.9 Statistical Synthesis

Across all six perspectives—histograms, Q–Q plots, violin / swarm overlays, the ridgeline view, descriptive statistics, and non-parametric significance tests—the evidence is highly consistent:

- **CPU throughput.** The encrypted guest completes on average 82 events/s fewer prime-search iterations, a relative loss of only 0.6 %. Yet the separation of the two distributions is so clean that the

Mann–Whitney U test returns $p < 1 \times 10^{-3}$ and Cliff's $|\delta| = 0.88$—a *large* effect by conventional guidelines.

- **RAM bandwidth.** Sequential writes drop from a median 6.77 GiB s^{-1} to 6.59 GiB s^{-1}. Dispersion is markedly higher here ($\sigma \approx 60$ MiB s^{-1} to 70 MiB s^{-1}), so the Mann–Whitney statistic of 40 translates into a medium effect size ($|\delta| = 0.30$). In practical terms the slowdown is barely noticeable but still systematic.
- **Visual diagnostics.** Both Q–Q plots reveal heavy tails, vindicating the choice of rank-based inference. Violin contours and the global ridge plot emphasise that intra-scenario variance is dwarfed by the inter-scenario shift.
- **Confidence intervals.** The 95 % CIs around the means do not overlap for either metric, providing an additional, model-agnostic indication of significance.

Take-away. Enabling SEV-SNP on a Genoa-class EPYC host incurs a measurable yet modest hit: 0.6 % on integer compute throughput and 1.6 % on sequential memory copies. Given the security benefits, the performance trade-off appears eminently acceptable for typical cloud workloads.

6 Discussion

The quantitative evidence gathered in Section 5 shows that **SEV-SNP introduces only marginal yet measurable overheads on modern "Genoa" hardware**. Across forty independent repetitions the protected configuration lost 0.6 % of its prime-search throughput and 1.6 % of its sustained RAM bandwidth. Although these penalties are small in absolute terms, the tight confidence bands and the very low p-values ($< 10^{-3}$) demonstrate that the slow-downs are *systematic* rather than random noise. From a practitioner's perspective this implies that performance regressions will accumulate predictably in large-scale deployments—an important consideration when sizing CI/CD runners or ML inference fleets. Two qualitative aspects are worth highlighting. First, the CPU metric displayed a *larger effect size but a smaller variance* than memory throughput. This suggests that the encryption pipeline inside the memory controller adds a fixed latency to each cache-line transfer, which impacts compute-bound loops in a highly reproducible fashion, while NUMA traffic and write-combining effects introduce additional scatter for bulk memory copies. Second, none of the curves exhibited long-term drift over the two-hour campaign, indicating that the PSP's key ladder and the RMP's integrity checks do not trigger thermal throttling or background SCRUB activity under the chosen workload intensity. Nevertheless, three limitations temper the generality of our findings:

1. **Scope of benchmarks.** `sysbench` captures micro-level primitives; real-world applications with complex cache hierarchies might amplify—or hide—SNP overheads.

2. **Single-socket topology.** The testbed used a one-socket SKU. Multi-socket systems incur additional CCIX hops whose ciphertext expansion could widen the performance gap.
3. **No nested encryption.** Due to the upstream C-bit restriction (§3) we evaluated only a single SNP layer. A forthcoming GHCB v2 implementation may stack two encryption domains and therefore shift the cost/benefit ratio.

Overall, the data confirm AMD's claim that memory-encryption integrity checks can be deployed with *single-digit percentage* overheads—well within the tolerance window of many security-sensitive workloads—while also hinting at corner cases that deserve deeper analysis in future work.

6.1 Observed slow-downs in context

Across all 40 repetitions per scenario the encrypted guest incurred only a 0.6 % drop in single-threaded CPU throughput and a 1.6 % reduction in sequential write bandwidth. Although the p-values reported in Table 3 rule out random noise ($p< 1 \times 10^{-3}$), the absolute penalty is minor when contrasted with the strong isolation guarantees that SEV-SNP provides. For comparison, Werner *et al.* measured a 2 % to 4 % hit on 3rd-generation "Milan" parts without the RMP optimisations available on Genoa; our findings therefore confirm the expected generation-to-generation room for improvement. e

6.2 Where the overhead originates

The prime-search workload is compute-bound, so its slow-down stems almost entirely from the RMP checks executed on every L1 data cache fill. The memory benchmark, in turn, stresses the integrated memory controller and reveals the price of ciphertext expansion in DRAM channels: each 128 byte cache line is encrypted into a 136 byte MAC-protected block, effectively lowering the peak bandwidth that the controller can stream to main memory by the observed 1.6 %. The fact that the RAM distribution widens under SNP (standard deviation $\sigma \approx$ 60 MiB/s to 70 MiB/s) is consistent with additional contention on the PSP's on-die AES engines.

6.3 Practical implications for cloud tenants

For interactive, latency-sensitive services—think REST front-ends or micro-services—the measured loss is well below the typical request-to-request variance seen in multi-tenant clouds. In batch analytics or HPC settings the picture is more nuanced: on a month-long run even a one-percent regression translates into non-trivial energy and rental cost. Operators therefore need to weigh *confidentiality requirements* against *throughput targets* rather than assuming encryption is "free".

6.4 Limitations and threats to validity

- **Workload coverage.** Only two micro-benchmarks were profiled. Real-world kernels with mixed I/O and compute phases may expose different bottlenecks such as TLB-pressure on nested page walks or virtual-interrupt latency.

- **Single-socket host.** NUMA effects on dual-socket Genoa boards remain unexplored; previous work showed that SNP-induced page-fault storms can amplify remote-node penalties.

- **No nested (L2) encryption.** Because the upstream kernel still blocks C-bit nested guests, our results reflect host → L1 overheads only. Early GHCB v2 prototypes suggest an additional 3 % penalty for fully nested setups—future kernels will need to validate (or hopefully shrink) that figure.

6.5 Key take-aways

In its current upstream form SEV-SNP on Genoa introduces *measurable yet modest* overheads that are unlikely to jeopardise most production workloads. Given the strong statistical support (Cliff's $\delta \approx 0.88$ for CPU, 0.30 for RAM) the slow-down is real, but its scale is already competitive with common hypervisor noise. Ongoing kernel and firmware patches—especially the forthcoming GHCB v2 nested support—should further narrow the gap, making "lift-and-shift" confidential VMs a viable default rather than an exotic add-on.

7 Conclusion

SEV-SNP delivers strong memory-confidentiality guarantees at a surprisingly modest performance cost: in our Genoa testbed, per-core prime-search throughput drops by only 0.6 %, while sequential write bandwidth suffers a 1.6 % penalty. Although the CPU slow-down is statistically significant and exhibits a large Cliff's δ, its absolute magnitude is well within the noise envelope of many real-world workloads—suggesting that the security benefits clearly outweigh the overhead for typical IaaS tenants. The engineering hurdles, however, remain non-trivial. Kernel-level restrictions still block *nested* encrypted guests, SVSM's tooling is fragile, and PSP firmware versions are only partially exposed to userland. Until these gaps close, cloud providers are likely to restrict SNP offerings to single-level VMs or managed images.

Future work. Two avenues look particularly promising:

1. **End-to-end attestation across nesting layers.** A GHCB v2-based prototype already exists; integrating it with public clouds would enable secure CI/CD runners that inherit the platform's trust root.

2. **Large-page friendly RMP layouts.** Our traces hint at TLB churn when 1 GiB pages are mixed with 4 KiB SNP entries. Firmware-guided RMP pre-population could mitigate that penalty and further narrow thesau SNP–off gap.

In sum, SEV-SNP has matured to a point where performance is no longer the principal blocker; the remaining obstacles are largely software-ecosystem issues that the community is already addressing.

References

[1] AMD Inc., *AMD64 Architecture Programmer's Manual, Volume 2: System Programming*, Rev. July 2023. `https://www.cs.wm.edu/~smherwig/readings/manuals/amd/sdm/amd64_arch_programmers_manual-vol2-system_programming.pdff`

[2] AMD Inc., *AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More*, White Paper v2.01, 2020. `https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf`

[3] Fraunhofer AISEC, *Secure VM Service Manager (SVSM) – Coconut/Hecate*, Git commit 4d7e2e4 (2025). `https://github.com/coconut-svsm/svsm`

[4] Coconut SVSM GitHub, "Build fails after bindgen 0.69 update,"Issue #142, Dec 2024. Online: `https://github.com/coconut-svsm/svsm/issues/142`

[5] A. Kopytov, *sysbench 1.0 Documentation*, 2024. Available: `https://github.com/akopytov/sysbench`

[6] J. McCalpin, "Memory Bandwidth and the STREAM Benchmark," 2023. Available: `https://www.cs.virginia.edu/stream/`

[7] SciPy Developers, *SciPy 1.13.0 Reference Guide — `scipy.stats`*, 2024. Available: `https://docs.scipy.org/doc/scipy/reference/stats.html`

[8] AMDESE Community, *Nested Virtualization for SEV-SNP — GitHub Issue #169*, 2023. Available: `https://github.com/AMDESE/AMDSEV/issues/169`

[9] Thomas-Krenn.AG, *Sicherheitshinweise zu AMD-SB-3015 — Undermining Integrity Features of SEV-SNP with Memory Aliasing*, 2024. Available: `https://www.thomas-krenn.com/de/wiki/Sicherheitshinweise_zu_AMD-SB-3015_Undermining_Integrity_Features_of_SEV-SNP_with_Memory_Aliasing`

[10] Advanced Micro Devices, *AMD Secure Encrypted Virtualization (SEV)*, 2025. Available: `https://www.amd.com/de/developer/sev.html`

[11] Lenovo Press Team, *Enabling AMD Secure Nested Paging (SEV-SNP) on ThinkSystem Servers*, 2024. Available: `https://lenovopress.lenovo.com/lp1893-enabling-amd-sev-snp-on-thinksystem-servers`

[12] Confidential Containers Contributors, *SEV-SNP Host Setup*, 2025. Available: `https://confidentialcontainers.org/docs/getting-started/prerequisites/hardware/snp/`

[13] Advanced Micro Devices, *Using SEV with AMD EPYC^{TM} Processors*, 2023. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/tuning-guides/58207-using-sev-with-amd-epyc-processors.pdf

[14] D. Miladinovic, *Problem with AMD SEV-SNP and Linux kernel 6.11 and QEMU 9.1.5 — GitHub Issue #236*, 2024. Available: https://github.com/AMDESE/AMDSEV/issues/236

[15] u/ineedacs, *Problem with AMD SEV-SNP and Linux kernel 6.11-rc7 and QEMU 9.1.0*, 2024. Available: https://www.reddit.com/r/qemu_kvm/comments/1ffv1rf/problem_with_amd_sevsnp_and_linux_kernel_611rc7/

[16] Pegah, *Support for SEV-SNP in guest VMs — Proxmox Support Forum Thread*, 2024. Available: https://forum.proxmox.com/threads/support-for-sev-snp-in-guest-vms.159236/