The Price of Privacy: A Performance Study of Confidential Virtual Machines for Database Systems

Lina Qiu
Boston University, USA
qlina@bu.edu

Rebecca Taft
Cockroach Labs, USA
becca@cockroachlabs.com

Alexander Shraer
shralex@gmail.com

George Kollios
Boston University, USA
gkollios@bu.edu

ABSTRACT

Confidential virtual machines (CVM) use trusted hardware to encrypt data being processed in memory to prevent unauthorized access. Applications can be migrated to CVM without changes, i.e., lift and shift, to handle sensitive workloads securely in public clouds. AMD Secure Encrypted Virtualization (SEV) is one of the prominent technologies that provides hardware support for CVM. In this paper, we investigate various system operations, including CPU, memory, and disk and network I/O, to understand the performance overheads of SEV-supported CVMs. Our findings indicate that memory and I/O-intensive workloads can incur significant overhead. We then study the performance implications of running unmodified database applications, specifically CockroachDB, on CVMs by examining typical data access patterns of OLTP and OLAP workloads. A notable performance overhead of up to 18% is observed for TPC-C workload running on multinode database clusters, and an overhead of up to 13% is observed for TPC-H workload running on single-node database instances. The non-negligible overhead suggests the potential and necessity for database optimizations with respect to CVM, particularly for time-sensitive workloads. We offer a glimpse of the effect that CVM overhead can have in query planning using a simple join query: the optimal join algorithm becomes suboptimal on CVM, along with discussion of potential optimizations for reducing CVM overhead in the realm of database applications.

ACM Reference Format:
Lina Qiu, Rebecca Taft, Alexander Shraer, and George Kollios. 2024. The Price of Privacy: A Performance Study of Confidential Virtual Machines for Database Systems. In *20th International Workshop on Data Management on New Hardware (DaMoN '24), June 10, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3662010.3663440

1 INTRODUCTION

Adopting public clouds is a cost-effective solution for organizations since they can forego the resource management process and only pay for the resources they utilize. For sensitive workloads like financial transactions and medical records, security requirements are the main concern before organizations adopt a cloud solution. Cloud providers offer various solutions to protect data confidentiality, integrity, and authenticity, such as secure channels for data-in-transit (on the network) and storage encryption for data-at-rest (on disks). However, data-in-use (being processed in memory) until recently has lacked effective protection: it was processed in plaintext unless advanced and computationally expensive cryptographic strategies, such as homomorphic encryption, were applied. Entities who gain access to the memory, e.g., cloud service providers and malicious tenants on the same host, can freely view the data. End-to-end encryption was offered by some cloud services but precluded meaningful computation or processing of the data in the cloud.

Confidential Computing (CC) is an emergent technology that aims to enforce sensitive data protection during processing so that sensitive data owners can safely outsource their data and computation with end-to-end protection in public clouds. CC achieves data-in-use protection by isolating sensitive data within trusted execution environments (TEE). Untrusted code running outside TEE is unable to view or access data inside TEE, even if it is running at the highest privilege levels (e.g., the hypervisor). Compared to homomorphic encryption, CC is less computationally expensive and does not have restrictions on the range of supported operations because the computation inside TEE is performed directly on plaintext data. CC introduces a new methodology for implementing end-to-end encryption, while allowing arbitrary processing of plaintext data inside TEE. Secure enclaves and confidential virtual machines are the two continuously evolving hardware-assisted TEE implementations. Intel SGX [35] offers the ability to create isolated and encrypted regions of memory called enclaves, within which sensitive code and data are protected from processes outside of the enclaves. Given the restricted isolation level, enclaves have a narrow attack surface exposed to host platforms. However, they have some limitations that make the adoption more challenging. For example, SGX SDK only offers C/C++ and Rust environments, and developers need to be very skilled for effective workload partitioning, since SGX requires splitting an application into "enclave" and "host" that are responsible for sensitive workload processing and non-sensitive functions such as basic networking and file I/O respectively. Confidential virtual machines (CVM), on the other hand, encrypt data loaded in memory with the VM-specific key and provide VM-level isolation. Technologies that support CVM include AMD SEV [26, 38] and Intel TDX [24]. The main advantage of CVM, compared to secure enclaves, is that applications can run on CVM without code changes (lift and shift).

Given the straightforward lift-and-shift software migration to CVM and the appealing security features that protect data-in-use, understanding the performance of CVM is important for a broader adoption of this technology. Existing results indicate that the performance overhead of CVM can be small, typically < 6% for some workloads [3–5], such as the Black-Scholes model. However, there are instances [25, 32, 47] where the overhead can be substantial and exceeds 20%. Despite the prior research efforts, the performance implications of running collaborative transactional workloads, such as TPC-C, and analytical workloads, such as TPC-H, using unmodified database instances on CVMs remain insufficiently comprehended.

In this paper, we study the performance implications of running OLTP and OLAP workloads using unmodified Database Systems on CVMs and confidential clusters supported by AMD SEV. Confidential clusters built out of CVMs is a promising solution to serve the global user base with high-performance, high-availability and fault-tolerant transactions while fulfilling the security requirements. The investigation is done with CockroachDB (CRDB) – a scalable Distributed SQL DBMS that was built from the ground up to support global workloads while maintaining high availability and strong consistency. It is resilient to disk, machine, rack, and even datacenter failures through replication and automatic recovery mechanisms [43, 46]. Our findings indicate that the overhead of employing CVM is not significant for CPU-intensive workloads, but it can accumulate to a significant amount for memory-intensive workloads, I/O-intensive workloads, and more complex workloads like OLTP and OLAP. In summary, the paper makes the following contributions:

- A thorough analysis for the causes of CVM performance overhead based on publicly available documentation of AMD SEV [26, 38]. We also conduct a series of microbenchmarks to support the analysis.
- A performance study of typical data access patterns of OLTP and OLAP workloads on CVMs and confidential clusters. Non-negligible overhead is observed comparing to regular VMs (RVM).
- In the performance study, we examine the individual contributions of the causes of CVM performance overhead to the overall overhead, and then discuss potential avenues for optimizations, particularly improving cache performance.

The remainder of the paper is organized as follows. In Sec. 2, we provide an overview of the designs of the key security features of AMD SEV and identify the causes of performance overhead. In Sec. 3, we describe the threat model of CVMs and connecting them as a confidential cluster. The setup of our experimental evaluation is outlined in Sec. 4, along with the definitions of performance metrics. Next in Sec. 5 we present a series of microbenchmarks to empirically support the analysis for the causes of CVM performance overhead. In Sec. 6, we first show the negative performance impact of SEV on multinode database clusters that run OLTP workloads, and then show the impact on single-node database instances that run TPC-H workload. Lastly, related work is covered in Sec. 7, followed by conclusions in Sec. 8.

2 AMD SEV TECHNOLOGIES

AMD SEV [26] is primarily designed to protect the confidentiality of data in memory for isolated VMs. Data loaded in the private memory of a VM is encrypted by the dedicated VM key generated and managed by the trusted hardware. The state-of-the-art generation of SEV, called SEV-SNP [38], builds upon existing SEV functionality and adds strong memory integrity protection, enabling more secure cloud computing. In the framework of SEV, the AMD System-On-Chip (SOC) hardware, the AMD Secure Processor (AMD-SP), and the CVM itself are fully trusted components, while all other CPU software components and PCI devices are untrusted. AMD-SP can provide attestation for remote parities to verify the trusted states of a CVM. CVM separates its memory into private and shared memory. Private memory is accessible only by the CVM, whereas shared memory is also accessible to untrusted host software.

Confidentiality is primarily accomplished using memory encryption. Each CVM is assigned a unique random encryption key generated by the hardware at boot time. The encryption key is stored in dedicated hardware registers where software cannot directly read it. When data from private memory leaves or enters the SOC, it is encrypted or decrypted by the hardware with the dedicated VM key. In other words, data in private memory is restricted to the VM it is intended for. The communication of a CVM with outside entities via I/O is done using a section of shared and unencrypted memory called SWIOTLB, which currently has a default size of 64MB in Linux kernel [16]. That is, the CVM must take its outgoing data, which is marked as private and encrypted with the dedicated VM key, and copy it into shared memory pages. I/O devices, such as network interface cards and storage devices, then take the data from the shared memory pages. Since the shared memory pages are not encrypted with the dedicated VM key, appropriate software encryption protocols like TLS and Full Disk Encryption (FDE) must be used for the security of I/O traffic. Similarly, any incoming data is first placed into shared memory pages and then copied to the private memory of the CVM. This method is called bounce buffering.

Similar to confidentiality guarantees, integrity guarantees added to SEV-SNP only apply to private memory. Integrity is enforced by Reverse Map Table (RMP), which is a single data structure shared across the system that tracks the owner of each page of memory. An RMP check is performed at the end of a translation from a virtual address to a physical address, to ensure that the memory page is being accessed by an authorized entity. All write memory accesses in the system with Reverse Map Table, including those from SEV-SNP VMs, from the hypervisor, and from non-SEV VMs, require RMP checks. Read accesses from SEV-SNP VMs to private memory pages require RMP checks, but read accesses from the hypervisor and non-SEV VMs do not require RMP checks, since data confidentiality is already protected via memory encryption.

From the designs of SEV, we find three differences between SEV CVM and non-SEV VM that are essential to the fulfillment of security guarantees. These differences are the causes of performance overhead: 1) cross-boundary encryption/decryption, which takes place when private data leaves or enters the SOC; 2) I/O bounce buffering; 3) RMP checks for read memory accesses. Cross-boundary encryption/decryption applies to all SEV technologies.

The requirement for I/O bounce buffering will be resolved in the next-generation of SEV technology. SEV Trusted I/O (SEV-TIO) [39] allows extending VM's trust boundaries to include devices, such that the VM and trusted devices can interact directly in private memory. In other words, I/O devices can be included in the trust boundaries and the data will no longer need to bounce to and from the shared memory buffer. RMP check is specific to SEV-SNP. It is not included in the following experiments, since SEV-SNP on Google Cloud Platform was in the preview stage at the time of experiments.

3 THREAT MODEL

On a host server, the only trusted components are the CVMs owned by the client, AMD SOC, and AMD-SP. All other software components and devices are assumed to be adversarial, including the hypervisor and other VMs on the same host. The adversarial components may conspire to compromise the confidentiality and integrity guarantees of SEV. Authorized CVM operators who have login access to the the CVMs, in addition to the client, can access the private data of the CVMs. Note that SEV CVM cannot guarantee availability. Denial-of-Service (DoS) and side-channel attacks [8, 10, 27, 31, 37] are orthogonal to this paper.

A confidential cluster is modeled as a collection of CVMs in one coherent and verifiable confidential context. In addition to the requirements to fulfill end-to-end protection of data confidentiality and integrity, a confidential cluster requires attested connections between CVMs, such that sensitive workloads are guaranteed not being exposed to any unauthenticated entity. Specifically, existing CVMs in a confidential cluster should be able to verify that newly joined CVMs are in trusted states when the cluster scales up for heavier workloads or some nodes fail and re-join. The client, on the other hand, should be able to verify all the security features (confidentiality, integrity and authenticity) of a confidential cluster in a single hardware-rooted certificate at any time [41]. Adversaries may attack CVM attestation and counterfeit TEE to collect sensitive information.

4 EXPERIMENTAL SETUP AND METRICS

All experiments are done on Google Cloud Platform n2d-standard-16 machines (16 vCPU, 32MB L3 cache, 64GB memory) based on 3rd Gen AMD EPYC processors (Milan). CVM has confidential computing (SEV) enabled and regular VM (RVM) does not. We did not employ SEV-SNP since it was in the priview stage at the time of our experiments. For disk and network I/O, CVM uses the default 64MB SWIOTLB [16] to communication with outside entities. It is sufficient for all our experiments without running into 'swiotlb buffer is full' error. In other words, the size of SWIOTLB does not contribute to the I/O performance degradation in our experiments. The remaining memory in CVM is encrypted and marked as private. All machines use Ubuntu 20.04 with Linux kernel 5.15 and gVNIC network interface. Unless otherwise specified, each machine is equipped with 500 GB performance persistent SSD disk (pd-ssd). A confidential cluster consists of three interconnected CVMs. The version of CockroachDB is v23.1.1. Each CockroachDB node is allocated with 16GB memory size (25% of the total memory) and manages its own 128MB memory cache, which is the default setting.

We use latency and throughput as the performance metrics in the experiments. The reported benchmarking results are the average of at least 5 runs. Let R_l and C_l denote the measured latency for RVM and CVM respectively. The latency overhead is defined as $\frac{C_l - R_l}{R_l}$. Similarly, let R_t and C_t denote the measured throughput for RVM and CVM respectively. The throughput overhead is defined as $\frac{R_t - C_t}{R_t}$.

5 MICROBENCHMARKS

The microbenchmarks are presented following the computer memory hierarchy. Non-negligible overhead is observed in memory read/write operations and disk and network I/O operations, which correspond to the first two identified causes of CVM performance overhead in Sec. 2, i.e., cross-boundary encryption/decryption and I/O bounce buffering, respectively.

5.1 Sysbench CPU performance

CVM and RVM should have the same CPU performance if the data is internal in the CPU and does not leave the SOC all the time. We use Sysbench [28] CPU workload, which performs prime number validation by doing standard division of the number by all integers between 2 and the square root of the number, to evaluate the performance of CPU-intensive tasks. The CPU workload is run with a single thread for 300 seconds. Table 1 compares the average events per second between RVM and CVM. The worst-case overhead is less than 0.2%. CVM does not experience significant CPU performance overheads compared to RVM.

cpu-max-prime	5000	10000	20000	50000
RVM (events/s)	9345	3668	1440	416.51
CVM (events/s)	9328	3661	1439	416.01
Overhead	0.18%	0.19%	0.06%	0.12%

Table 1: Sysbench CPU performance

5.2 Sysbench memory performance

We use Sysbench memory workload to evaluate the throughput of memory read/write under different access patterns. Sysbench memory tests allocate a memory buffer and then read or write in it until the entire buffer has been read from or written to. This is repeated until the provided total data volume is reached.

The memory buffer size is configured to 1MB and 64MB. A n2d-standard-16 machine has 32MB L3 cache, hence a large portion of the 1MB memory buffer should be cached in CPU, while the 64MB memory buffer requires more cache swaps in the execution. Fig. 1 shows the overheads of CVM's memory throughput. CVM exhibits less than 1% overheads with 1MB memory buffer. The overheads are more significant for 64MB memory buffer. Sequential read/write experience about 5% overhead and random read/write experience about 14% overhead. The difference between the overheads of the two buffer size configurations indicates that the performance overhead of SEV is affected by cache performance. The observation is consistent with the analysis of cross-boundary encryption/decryption overhead based on the designs of AMD SEV. A

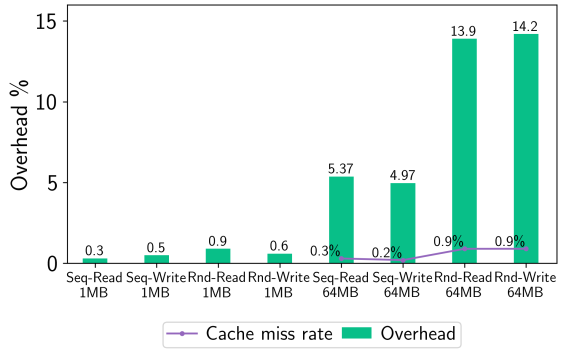

Figure 1: Sysbench memory performance

higher cache miss rate implies more frequent cross-boundary (AMD SOC) data transitions and therefore higher encryption/decryption overhead. To further examine this observation, we use Cachegrind [45] to measure cache miss rates of the tested workloads. Cachegrind is a high-precision tracing profiler that can simulate how a program interacts with a machine's cache hierarchy. The magnitude of the cache miss rates provides us some insights of the cache performance of the workloads, but the numbers are off by default because the cache simulations are basic and do not reflect the behaviour of a modern machine [45]. Profilers like perf [33] that access hardware counters can provide actual cache information but are not supported by most of virtual machines, including n2d-standard-16.

The cache miss rates reported by Cachegrind for workloads with 1MB memory buffer are negligible and very close to zero, because Cachegrind makes the simple assumption that the entire 1MB memory buffer can be cached in the processor. The negligible cache miss rates explain the negligible overheads in 1MB memory buffer throughput tests. For the 64 MB memory buffer setting, overheads are proportional to the simulated cache miss rates, see Fig. 1. The fact that random workloads experience higher overhead than sequential workloads also indicates the strong correlation between cache performance and the overhead of cross-boundary encryption/decryption.

5.3 FIO disk performance

In this microbenchmark, each test VM is attached with either performance persistent SSD disk (pd-ssd), or local SSD disk with NVMe interface. CVM uses NVMe interface for pd-ssd, but RVM uses SCSI interface for pd-ssd [12]. Local SSD disks can only provide temporary storage and are physically attached to the server that hosts the VM. Therefore, local SSD disks offer superior I/O operations per second (IOPS) and throughput compared to persistent SSD disks.

Following the instructions in [13], we use Flexible I/O tester (FIO) [7] to first examine the sequential (throughput) and random (IOPS) read/write performance of 500GB and 2500GB pd-ssd disks attached to n2d-standard-16 VMs, see Table 2. Throughput is tested with 16 parallel streams using 1MB I/O block size and 64 I/O depth. IOPS is tested with 4KB I/O blocks and 256 I/O depth. Persistent disk performance scales with the size of the disk and with the number of vCPUs until it reaches either the limits of the disk or the limits of the VM. For a 500GB pd-ssd disk, both RVM and CVM can meet the disk performance limits. For a 2500GB pd-ssd disk, both RVM and CVM reach the limits of the VM. No overhead is observed in CVM's pd-ssd disk performance evaluation due to the caps of maximum performance limits [15]. The disk I/O overhead of CVM is revealed in the tests of local SSD disks performance [14], see Table 3. We equip n2d-standard-16 machines with the maximum allowable number of local disks, i.e., 24 local disks, each with a size of 375 GB. For CVM, FIO reports 827k read IOPS and 757k write IOPS, which is about ***half*** of the performance of RVM. The difference in throughput performance for sequential read/write between RVM and CVM is minor since they all reach the maximum performance limits of 24 local SSD disks [11].

VM-Disk	R Throughput	R IOPS	W Throughput	W IOPS
RVM-500GB	483MiB/s	21k	483MiB/s	21k
CVM-500GB	483MiB/s	21k	483MiB/s	21k
500GB pd-ssd limits [15]	480MiB/s	21k	480MiB/s	21k
RVM-2500GB	1205MiB/s	25k	1200MiB/s	25k
CVM-2500GB	1203MiB/s	25k	1198MiB/s	25k
2500GB pd-ssd limits [15]	1200MiB/s	25k	1200MiB/s	25k

Table 2: FIO persistent SSD performance

VM-Disk	R Throughput	R IOPS	W Throughput	W IOPS
RVM-24×375GB	9832MiB/s	1783k	5471MiB/s	1400k
CVM-24×375GB	9825MiB/s	***827k***	5468MiB/s	***757k***
Max Limits [11]	9360MiB/s	2400k	4680MiB/s	1200k

Table 3: FIO local SSD performance

5.4 Netperf network performance

Two n2d-standard-16 machines are set up in the same zone (us-central1-a), and in different regions (us-central1-a, us-east1-b), to measure network latency and throughput using netperf [22] TCP_RR and TCP_STREAM protocols. The intra-zone network latency between two CVMs exhibits more than 11% overhead compared to the latency between two intra-zone RVMs, with an overhead as substantial as 20% for P99 latency, see Fig. 2. As demonstrated in Fig. 3, the network throughput between two intra-zone CVMs exhibits 10% overhead. On the other hand, comparable network latency and throughput are observed in the cross-region setting with or without SEV enabled, as the long-distance transmission latency dominates the overall runtime, which makes the overhead less significant.

Analysis of CVM I/O. Due to the I/O bounce buffering via SWIOTLB, CVM experiences non-negligible overhead in disk and network performance. Traditionally, SWIOTLB was not performance critical, becuase it was introduced to resovlve the addressing problem of old (and slow) devices, such that DMA operations can be handled with some overhead. In the setup of CVM, all I/O has to go through SWIOTLB. It's worth noting that an inadequately sized SWIOTLB buffer will lead to more severe I/O performance degradation, but in our experiments, we did not run into this scenario. A series of optimizations regarding SWIOTLB are actively being studied to improve the I/O performance of CVM [1, 19].

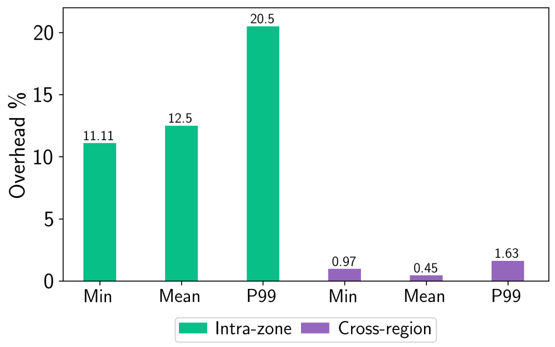

Figure 2: Netperf network latency

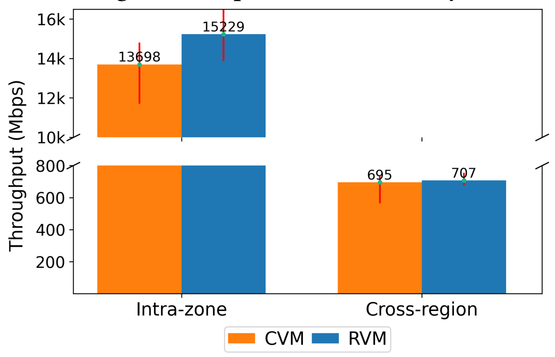

Figure 3: Netperf network throughput

6 MACROBENCHMARKS

From the microbenchmarks, we pin down the two operations, i.e., memory read/write and network I/O, that should contribute a considerable portion of the overhead in the following macrobenchmarks. Disk I/O is not included because the test VMs are equipped with pd-ssd disks rather than local SSD disks.

6.1 Cluster OLTP

OLTP workloads typically involve collaboratively processing a large number of short and interactive transactions in real time. To capture and analyze the performance distinctions arising from the deployment of SEV on multinode clusters, we run TPC-C and YCSB benchmarks on 3-node CRDB clusters. The three nodes are deployed either as three different n2d-standard-16 VMs in the same zone, or hosted in a single n2d-standard-48 VM (48 vCPU, 192GB memory) with each node assigned 16 vCPU and 16GB memory. The latter local 3-node cluster is used as a baseline in our YCSB experiments to differentiate the individual contributions of cross-boundary encryption/decryption and network I/O bounce buffering to the overall overhead.

TPC-C simulates a complex data warehouse environment with multiple users executing transactions concurrently. Transactions per minute (tpmC) measures the throughput of the system with the number of completed new-order transactions per minute. Each type of transaction is constrained with a P90 response time requirement in the order of seconds, which establishes the passing criteria of the workload. We set the number of active connections at the load generator as equal to 4x the number of vCPUs, as per CockroachDB's production guidelines. Note that TPC-C is constrained to a maximum amount of throughput per warehouse (12.86 tpmC). To increase the throughput, more warehouses must be configured. Fig. 4 captures the growth of tpmC with the number of warehouses. The max tpmC of regular distributed clusters is 32370 and the max tpmC of confidential distributed clusters is 28673, which exhibits 11.42% overhead. In addition, regular distributed clusters can support up to 2700 warehouses following the passing criteria, while confidential distributed clusters can only support up to 2400 warehouses. At 2700 warehouses, the tpmC of confidential distributed clusters is 17.81% smaller than regular distributed clusters.

YCSB [18] offers a means of benchmarking database efficiency in handling simple read and write-intensive operations. We populate the YCSB database with 10 million records. The resulting data volume is about 13GB. Fig. 5 shows the results of YCSB-C read-only workload and YCSB-A 50-read-50-update workload. Experiments on local clusters (3-node CRDB clusters in one n2d-standard-48 VM) do not include network I/O. We observe that distributed confidential clusters (3-node CRDB clusters consisting of three n2d-standard-16 CVMs) experience 6-8% throughput overhead and local confidential clusters experience 2-4% throughput overhead for the simple YCSB workloads. From the statistics reported by the cloud Ops Agent [17], the disk read/write performance (i.e., disk throughput and IOPS) of regular and confidential clusters is at the same magnitude in both distributed and local settings. Therefore, we can conclude that the overhead of local confidential clusters is mainly driven by cross-boundary encryption/decryption. The differences (about 4%) between the overheads of distributed and local clusters are mainly driven by network I/O bounce buffering.

The requirement for I/O bounce buffering will be resolved in the next-generation of SEV technology, i.e., SEV-TIO, but cross-boundary encryption/decryption is essential to all SEV technologies. In the next section, we examine the performance of a single-node database with analytical workloads, focusing on the impact of cross-boundary encryption/decryption.

6.2 Single node OLAP

TPC-H is the most widely used OLAP benchmark that includes a set of 22 complex queries comprising a variety of analytical query operators. We generate TPC-H data with scale factor 1 on a single CRDB node; the resulting data volume is about 1GB. In the single-node setting, network I/O is not involved in the execution, and the small scale factor is chosen to mitigate disk I/O (spilling to disks), such that the overhead is primarily incurred by cross-boundary encryption/decryption (i.e., cache misses). The 22 queries are sent to CRDB sequentially in a closed loop.

The average runtime of executing 22 queries sequentially is 121 seconds on CVM comparing to 112 seconds on RVM. Fig. 6 is a cost breakdown analysis of the 22 queries. We observe that some queries exhibit higher runtime overhead than the others: Q5, Q6, Q7, Q9, Q10, Q12, Q14, Q15, and Q20 experience more than 6% overhead, while other queries have less than 3% overhead. We observe overhead as high as 13.1% for Q9. The non-negligible overhead suggests the potential for query optimization with respect to SEV.

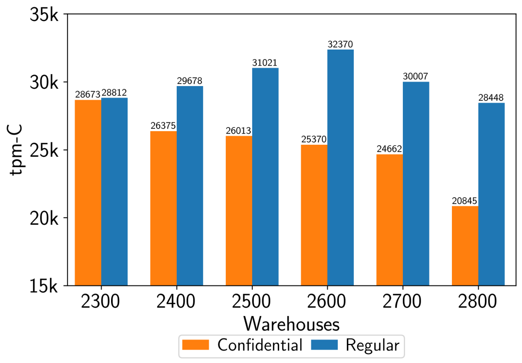

Figure 4: TPC-C Throughput

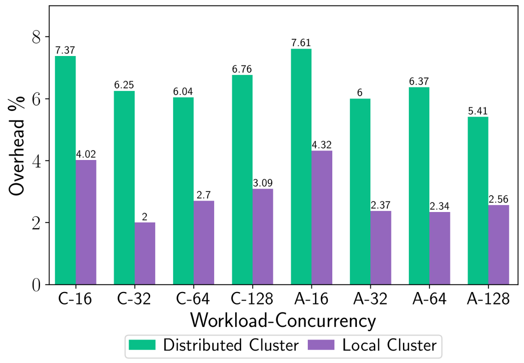

Figure 5: YCSB Throughput

The test VMs do not support using perf [33] to access hardware counters for actual cache performance. To check the connection between cache performance and query overhead, we use EXPLAIN ANALYZE on each TPC-H query in order to get the query plan with execution statistics. According to the statistics, most of the queries do not need disk usage except for the first step that loads the input from disks into the memory, unless otherwise specified. We find that the set of queries S_h = (Q5, Q6, Q7, Q9, Q10, Q12, Q14, Q15, Q20) with more than 6% overhead all have at least one join operator in their plans that requires $\geq$ 35MB working memory. The working memory statistics is indicated by the property "estimated max memory allocated" in the statement plan tree. Those queries with relatively low $<$ 3% overhead S_l = (Q3, Q4, Q8, Q11, Q16, Q17, Q18, Q19, Q21, Q22) have all the operators requiring no more than 16MB working memory. Considering the 32MB L3 cache size of the virtual machine, the execution of S_l operators in theory can be done in the processor with all real-time data in the cache. They require lower cache miss rates than the operators that need larger working memory for execution, and therefore less frequent cross-boundary data encryption/decryption and smaller query overhead. The remaining queries Q1, Q2, and Q13 also experience $<$ 3% overhead even though they all have one GroupBy (hash) operator in the plans requiring $\geq$ 35MB working memory. The GroupBy (hash) operators in the three queries require disk usage in addition to memory usage, which dominates the runtime of the operators and hides the impact of cross-boundary encryption/decryption.

Optimizations. Improving cache performance is critical to enhancing the overall efficiency of computer systems and has been a long-standing topic of research. Within the context of database optimization with respect to SEV, improving cache performance can reduce runtime overhead as well. We manually craft a query to offer a glimpse of the effect that SEV overhead can have on query planning. This example is an initial step in exploring potential avenues to reduce CVM memory encryption overhead for database applications.

The manually crafted query is a simple join between `lineitem` and `orders` tables on `orderkey` with filtering predicate `l_quantity < 6`, as shown in Fig. 7. In TPC-H, both `lineitem` and `orders` are indexed on the equality column `orderkey` and the indices have the same ordering. Merge join therefore has computational complexity $O(m + n + R)$, where m and n are the sizes of input tables and R is the size of join output. If we assume $O(1)$ hash lookup time, hash join also has $O(m + n + R)$ computational complexity, but its access pattern is more random than merge join due to the property of hashing. Therefore, hash join is expected to experience a higher cache miss rate and larger query overhead caused by cross-boundary data encryption/decryption.

In the experiments, we leverage join hints [30] to force the use of merge/hash join. The maximum amount of working memory that a processor can use during query execution is set to a large value (i.e., sql.distsql.temp_storage.workmem=10GB [29]), so that no disk spilling is triggered during query execution. The large memory budget allows hash join to buffer the hash table fully in memory, whereas merge join does not take advantage of the extra space and keeps generating the join by consuming the next batch of input from disk. The implementation makes hash join more runtime efficient than merge join in the large memory budget setting on RVM: 275ms for hash join comparing to 283ms for merge join. However, the efficiency reverses when SEV is enabled. Hash join has an average runtime of 309ms on CVM, a 12% overhead compared to RVM, whereas the average runtime of merge join only experiences a small increase (1.1%) to 286ms. This example shows that the overhead caused by SEV can affect the optimality of query plans. Potential avenues to reduce the memory encryption overhead include adjusting the query cost model to account for the cost of memory encryption and the design and implementation of cache-efficient data structures and algorithms. We leave such optimizations as future work.

7 RELATED WORK

Prior research efforts on the performance impact of hardware-based TEEs focus on either microbenchmarks for understanding the characteristics and pitfalls of a single TEE [9, 20, 21, 47] or a comparison between different TEEs [2, 23, 36]. Akram et al. [2] are the first to evaluate high-performance computing benchmarks on large SEV machines with multiple NUMA nodes. Yan and Gopalan [47] examine the performance overheads of CVM across three generations of AMD SEV using CPU, memory, and I/O benchmarks. In this paper, we have similar overhead results in the microbenchmark experiments. Unlike previous studies that target microbenchmarks in a single TEE, our work aims to show how the overheads affect

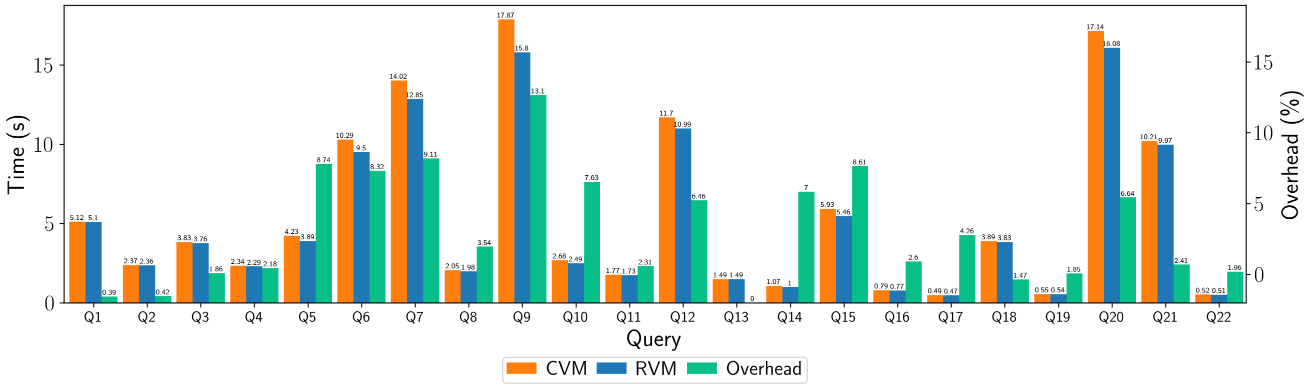

Figure 6: TPC-H SF 1 query runtime

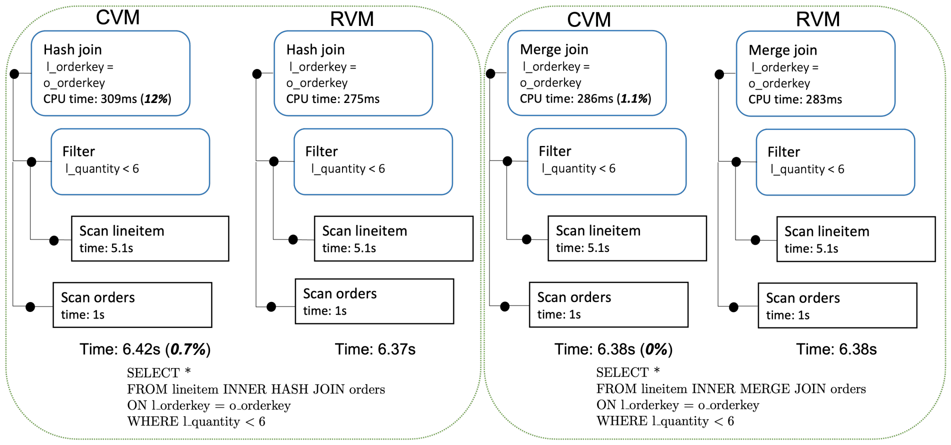

Figure 7: TPC-H SF 1 join plan comparison

the overall performance of an unmodified database system running across one or multiple CVMs. By introducing LibOSes into enclaves, [6, 40, 44] try to minimize the effort of refactoring for SGX. To improve the performance of TEE-based applications, a series of recent work introduce system-level [32, 42] or database-level optimizations [34] with respect to a specific TEE's architecture.

8 CONCLUSION

CVM is a rapidly evolving technology that bridges the gap of end-to-end data protection in public cloud environments without the need for application refactoring. This paper presents a thorough analysis for the performance overheads of SEV-supported CVM and shows how the overhead can lead to suboptimal strategies in database applications. For time-critical sensitive workloads, database-level performance optimization with respect to CVM is necessary for the adoption of cost-effective cloud solutions.

9 ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable feedback. This work was mainly done during Lina Qiu's internship at Cockroach Labs. The authors are grateful to Anoosh Saboori, Yahor Yuzefovich, Nathan VanBenschoten, Austen McClernon, Marcus Gartner, Tobias Grieger, and Xiang Gu for inspiring discussions and helpful suggestions.

REFERENCES

[1] 2022. *SWIOTLB performance optimizations.* https://lore.kernel.org/lkml/20220630024238.GA884@gao-cwp/T
[2] Ayaz Akram, Anna Giannakou, Venkatesh Akella, Jason Lowe-Power, and Sean Peisert. 2021. Performance analysis of scientific computing workloads on general purpose TEEs. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 1066–1076.
[3] AMD. 2020. *PostgreSQL on Google Compute Engine with N2D Confidential VMs.* https://www.amd.com/system/files/documents/amd-epyc7002-gcpn2d-confidential-postgresql.pdf
[4] AMD. 2021. *Microsoft Azure Confidential Computing Powered by 3rd Gen EPYC™ CPUs.* https://community.amd.com/t5/epyc-processors/microsoft-azure-confidential-computing-powered-by-3rd-gen-epyc/ba-p/497796
[5] AMD. 2022. *Google Cloud C2D VM Instances Powered by 3rd Gen EPYC.* https://www.amd.com/system/files/documents/3rd-gen-epyc-gcp-c2d-conf-compute-perf-brief.pdf
[6] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'keeffe, Mark L Stillwell, et al. 2016. {SCONE}: Secure linux containers with intel {SGX}. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 689–703.
[7] Jens Axboe. 2024. Flexible I/O Tester (FIO). https://git.kernel.dk/cgit/fio

[8] Dmytro Bogatov, Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2021. εpsolute: Efficiently Querying Databases While Providing Differential Privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2262–2276.

[9] Stefan Brenner, Michael Behlendorf, and Rüdiger Kapitza. 2018. Trusted execution, and the impact of security on performance. In *Proceedings of the 3rd Workshop on System Software for Trusted Execution*. 28–33.

[10] TH Hubert Chan, Kai-Min Chung, Bruce M Maggs, and Elaine Shi. 2019. Foundations of differentially oblivious algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2448–2467.

[11] Google Cloud. 2024. *About Local SSD disks*. https://cloud.google.com/compute/docs/disks/local-ssd#performance

[12] Google Cloud. 2024. *About Persistent Disk*. https://cloud.google.com/compute/docs/disks/persistent-disks

[13] Google Cloud. 2024. *Benchmark persistent disk performance on a Linux VM*. https://cloud.google.com/compute/docs/disks/benchmarking-pd-performance

[14] Google Cloud. 2024. *Benchmarking local SSD performance*. https://cloud.google.com/compute/docs/disks/benchmarking-local-ssd-performance

[15] Google Cloud. 2024. *Configure disks to meet performance requirements*. https://cloud.google.com/compute/docs/disks/performance

[16] Google Cloud. 2024. *Full SWIOTLB*. https://cloud.google.com/confidential-computing/confidential-vm/docs/troubleshoot-full-swiotlb

[17] Google Cloud. 2024. *Ops Agent metrics*. https://cloud.google.com/monitoring/api/metrics_opsagent#agent-disk

[18] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM symposium on Cloud computing*. 143–154.

[19] Jonathan Corbet. 2023. *A more dynamic software I/O TLB*. https://lwn.net/Articles/940973

[20] Tu Dinh Ngoc, Bao Bui, Stella Bitchebe, Alain Tchana, Valerio Schiavoni, Pascal Felber, and Daniel Hagimont. 2019. Everything you should know about Intel SGX performance on virtualized systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 1 (2019), 1–21.

[21] Muhammad El-Hindi, Tobias Ziegler, Matthias Heinrich, Adrian Lutsch, Zheguang Zhao, and Carsten Binnig. 2022. Benchmarking the second generation of intel SGX hardware. In *Proceedings of the 18th International Workshop on Data Management on New Hardware*. 1–8.

[22] Hewlett Packard Enterprise. 2021. *netperf*. https://github.com/HewlettPackard/netperf

[23] Christian Göttel, Rafael Pires, Isabelly Rocha, Sébastien Vaucher, Pascal Felber, Marcelo Pasin, and Valerio Schiavoni. 2018. Security, performance and energy trade-offs of hardware-assisted memory protection mechanisms. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 133–142.

[24] Intel. 2023. *Intel®Trust Domain Extensions*. https://cdrdv2.intel.com/v1/dl/getContent/690419

[25] Intel. 2023. *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

[26] David Kaplan, Jeremy Powell, and Tom Woller. 2016. AMD memory encryption. *White paper* (2016), 13.

[27] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'neill. 2016. Generic attacks on secure outsourced databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1329–1340.

[28] Alexey Kopytov. 2023. *Sysbench*. https://github.com/akopytov/sysbench

[29] Cockroach Labs. 2024. *Cluster Settings*. https://www.cockroachlabs.com/docs/stable/cluster-settings

[30] Cockroach Labs. 2024. *Join Hints*. https://www.cockroachlabs.com/docs/v23.2/cost-based-optimizer#join-hints

[31] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. 2018. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 297–314.

[32] Dingji Li, Zeyu Mi, Chenhui Ji, Yifan Tan, Binyu Zang, Haibing Guan, and Haibo Chen. 2023. Bifrost: Analysis and Optimization of Network {I/O} Tax in Confidential Virtual Machines. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. 1–15.

[33] Linux. 2024. *Perf: Linux profiling with performance counters*. https://perf.wiki.kernel.org/index.php/Main_Page

[34] Kajetan Maliszewski, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2023. Cracking-Like Join for Trusted Execution Environments. *Proceedings of the VLDB Endowment* 16, 9 (2023), 2330–2343.

[35] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. 2016. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*. 1–9.

[36] Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi. 2018. A comparison study of intel SGX and AMD memory encryption technology. In *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*. 1–8.

[37] Lina Qiu, Georgios Kellaris, Nikos Mamoulis, Kobbi Nissim, and George Kollios. 2023. Doquet: Differentially Oblivious Range and Join Queries with Private Data Structures. *Proceedings of the VLDB Endowment* 16, 13 (2023), 4160–4173.

[38] AMD Sev-Snp. 2020. Strengthening VM isolation with integrity protection and more. *White Paper, January* 53 (2020), 1450–1465.

[39] AMD SEV-TIO. 2023. Trusted I/O for secure encrypted virtualization. *AMD White Paper* (2023).

[40] Youren Shen, Hongliang Tian, Yu Chen, Kang Chen, Runji Wang, Yi Xu, Yubin Xia, and Shoumeng Yan. 2020. 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*. 955–970.

[41] Edgeless Systems. 2023. *Constellation security features*. https://docs.edgeless.systems/constellation/overview/confidential-kubernetes#constellation-security-features

[42] Meysam Taassori, Ali Shafiee, and Rajeev Balasubramonian. 2018. VAULT: Reducing paging overheads in SGX with efficient integrity verification structures. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. 665–678.

[43] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, et al. 2020. Cockroachdb: The resilient geo-distributed sql database. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*. 1493–1509.

[44] Chia-Che Tsai, Donald E Porter, and Mona Vij. 2017. {Graphene-SGX}: A practical library {OS} for unmodified applications on {SGX}. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. 645–658.

[45] Valgrind. 2023. *Cachegrind: a high-precision tracing profiler*. https://valgrind.org/docs/manual/cg-manual.html

[46] Nathan VanBenschoten, Arul Ajmani, Marcus Gartner, Andrei Matei, Aayush Shah, Irfan Sharif, Alexander Shraer, Adam Storm, Rebecca Taft, Oliver Tan, et al. 2022. Enabling the next generation of multi-region applications with cockroachdb. In *Proceedings of the 2022 International Conference on Management of Data*. 2312–2325.

[47] Mingjie Yan and Kartik Gopalan. 2023. Performance Overheads of Confidential Virtual Machines. In *2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 1–8.