**Performance challenges of memory protection with Confidential Computing**

Intel Confidential Computing (CC) technologies related to memory, UPI and CXL are designed to protect data in use with encryption and integrity checks. However, there are performance penalties an end user might experience depending on the nature of the workload. Based on our experiments confidential computing technologies like TDX and SGX that are supported by underlying features like Total Memory Encryption (TME), TME-bypass (TMEBP) and UPI-encryption, impact Trusted Execution Environments (TEEs) performance up to 3% for common use cases with TME, 20% with UPI-enc for use cases that involve remote memory accesses and 50% in really large guests with AI workloads Since top-tier hyperscalers intend to enable CC technologies ubiquitously in their cloud/datacenters, their KPI’s for TDX tend to be <5% generally for CPU/memory intensive workloads like SPEC CPU and as low as 1% for TMEBP. Hence based on our customer interactions, we believe these high overheads due to data encryption in SOC can directly impact Intel business as they impact several customer workloads including AI.

To understand the background, while TME and TME-MK (multi-key TME) provide full physical memory encryption of the DRAM, with single and multiple ephemeral keys respectively, UPI encryption protects cross-socket remote memory accesses. To minimize the impact of memory encryption on non-confidential guests, the SoC supports a feature called Intel TME bypass (TMEBP) Intel TDX and SGX built on top of TME-MK and UPI encryption, provides a hardware-isolated virtualization-based trusted execution environment (TEE). Depending on the workloads and use cases, a customer could deploy guests of different sizes, where size is generally determined based on the number of vCPUs and amount of memory. For optimal UMA and NUMA configurations, for instance on GNR-AP Intel supports SNC3 and HEX. Small guests are generally the most deployed in CSP environments.

We observe memory related performance challenges, some due to hardware/SOC and others related to software:

* Higher perf overhead due to TMEBP accesses to local memory due to SOC hardware. We observe around 3% performance impact due to TME bypass and ~5% due to TME for small guests. We attribute these overheads to Intel Memory Security Engine (MSE). Attached snap shot of gen-2-gen MSE overheads.
* Higher perf overhead for remote memory accesses due to TMEBP/TME/UPI-encryption related overheads in SOC hardware. While we expect local and remote memory overheads to be similar for TME and TMEBP, we observe them to be higher (~7%) in smaller guests. Additionally, TDX and SGX technologies enable protection of cross-socket traffic via UPI-encryption which is supported by UCE – UPI Cryptographic Engine. Encryption/Integrity overheads of UCE are an important factor. UPI-encryption can impact memory bandwidth performance up to 20%. A TEE that spans multiple CPU sockets tends to deploy applications that are NUMA aware. However, if they are not, they would be impacted due to this issue.
* TEE spanning over multiple-NUMA related performance drops:
  + Single socket: Significant performance overheads experienced by TEE’s when enabling sub-NUMA clusters like SNC, which is a customer desired feature due to low memory access latencies. Micro benchmarks like MLC show up to 20%, while AI workloads have shown even higher.
  + Multi-Socket: A TEE that spans multiple CPU sockets tends to deploy applications that are NUMA aware. In spite of NUMA awareness, they could be impacted due to this issue. We have observed up to 50% performance drops in AI workloads.

These are due to Linux VMM (KVM) related issues that impact both Intel and AMD based TEEs., and this impact is to both SGX and TDX.

The intent of the paper/presentation is general awareness of the performance issues and key factors behind them for consideration in future products, while we briefly mention potential workarounds/suggestions for our current roadmap products.

A graph with blue and orange lines

Description automatically generated