



CAGE: Complementing Arm CCA with GPU Extensions

Chenxu Wang^{1,2}, Fengwei Zhang¹, Yunjie Deng¹, Kevin Leach³,
Jiannong Cao², Zhenyu Ning⁴, Shoumeng Yan and Zhengyu He⁵

¹Southern University of Science and Technology, ²The Hong Kong Polytechnic University,
³Vanderbilt University, ⁴Hunan University, ⁵Ant Group



Confidential Computing



- An emerging concept and technique for data security
- Gradually attract cloud providers and third-party developers
 - Google Cloud, Microsoft Azure, Aliyun ...
- Hardware-assisted protection
 - Intel Trust Domain Extensions (TDX)
 - IBM Protected Execution Facility (PEF)
 - AMD Secure Encrypted Virtualization (SEV)
 - ...

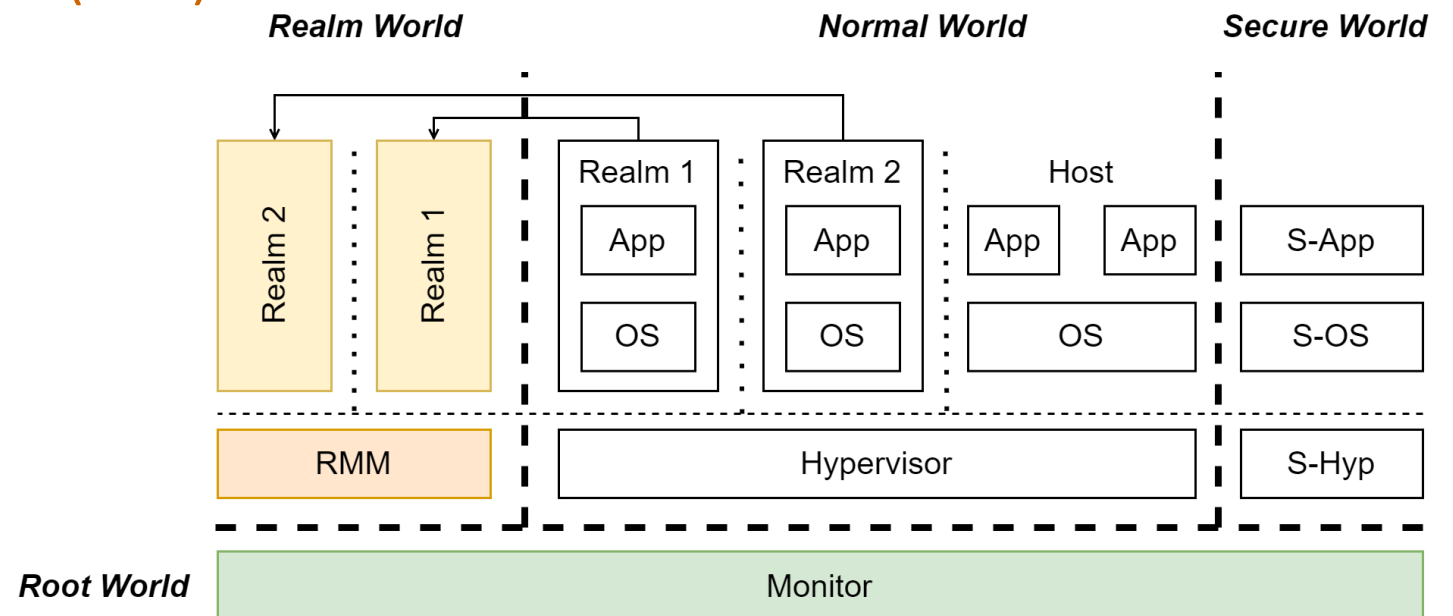


confidential computing

Arm Confidential Compute Architecture (CCA)



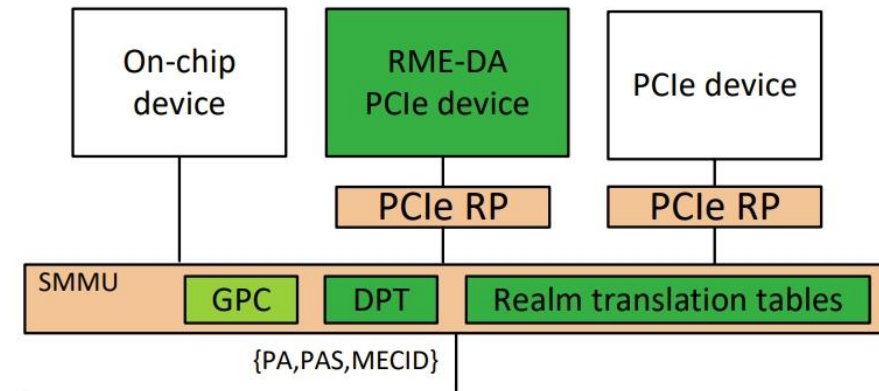
- Provide confidential computing for next-generation (Armv9.2) Arm devices
 - New security state for confidential computing: **Realm World**
 - Hardware-isolated **Root World**
 - New security supports
 - **Granule Protection Check (GPC)**
 - Memory encryption
- ...



Arm Confidential Computing Architecture (CCA)



- CCA is not completed: CCA on unified-memory GPUs
 - These on-chip GPUs are widely used in current Arm devices
 - But in Armv8 and early CCA, GPU is untrusted for Realms
- Arm introduces Device Assignment for Realm Management Extensions (**RME-DA**) to solve this problem, but ...
 - Still in the early stage
 - How it supports generic, on-chip GPUs is uncertain
 - No real-world hardware or software simulation



RME-DA focuses on managing PCIe-connected device. Source from Arm DEN0129 manual, version B.a.

Motivation & Goals



- Providing Arm CCA with confidential, unified-memory GPU computing support
 - Compatibility with Arm CCA
 - Strong data security
 - Low performance overhead
 - No hardware changes

Threat Model & Assumptions

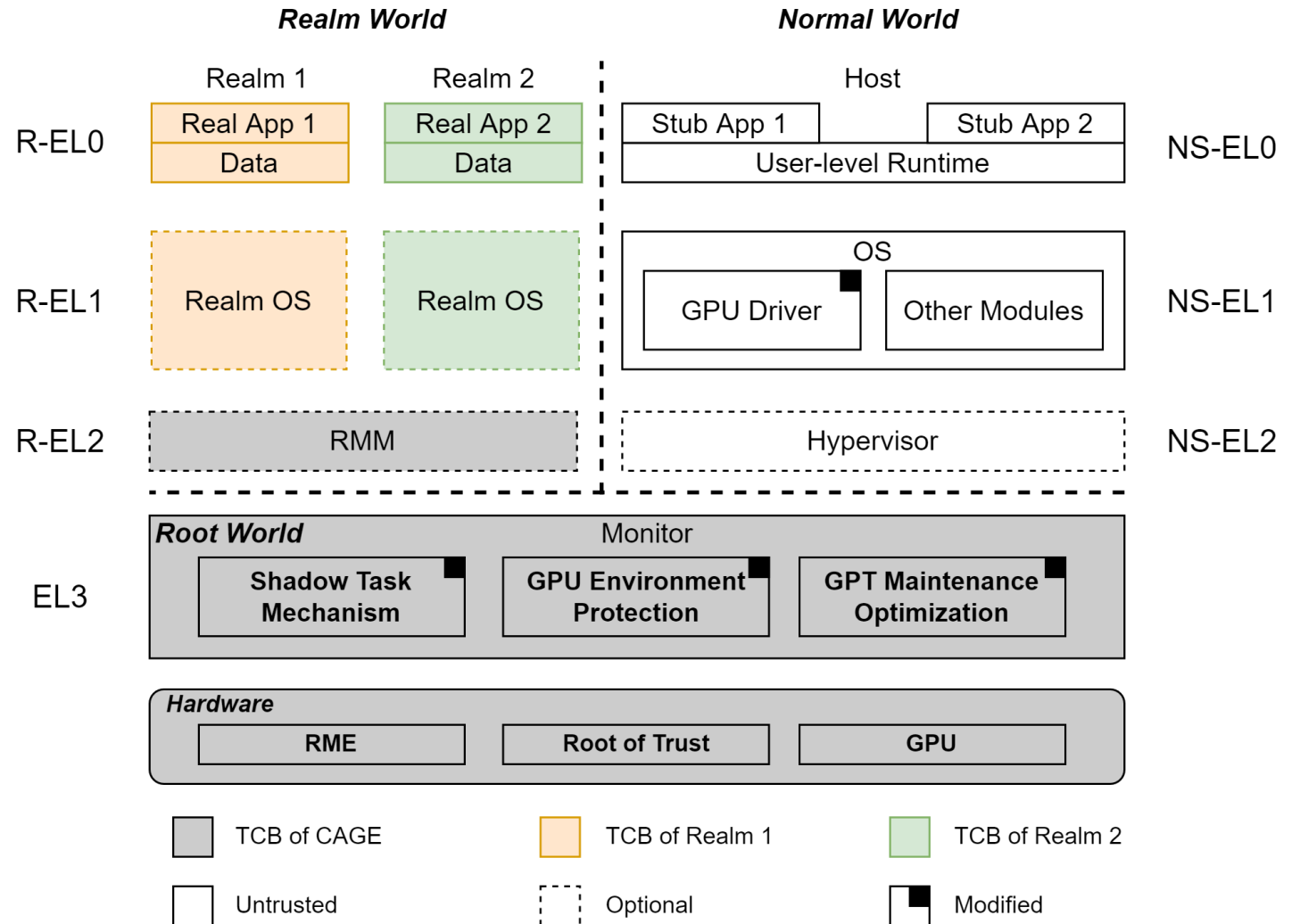


- Follow **Arm CCA**'s threat model
 - Software in Normal World and Secure World is untrusted for realms
 - Peripherals except GPU are untrusted
- Assume remote attestation and secure boot support
 - Trust existing CPU-side isolation firmware in Arm CCA (Monitor and RMM)
- Physical/side-channel/DoS attacks can be addressed by orthogonal works

Complementing Arm CCA with GPU Extensions (CAGE)



- Monitor
 - Security responsibilities
 - Three mechanisms
- User-level runtime & GPU driver
 - GPU functionality guarantee

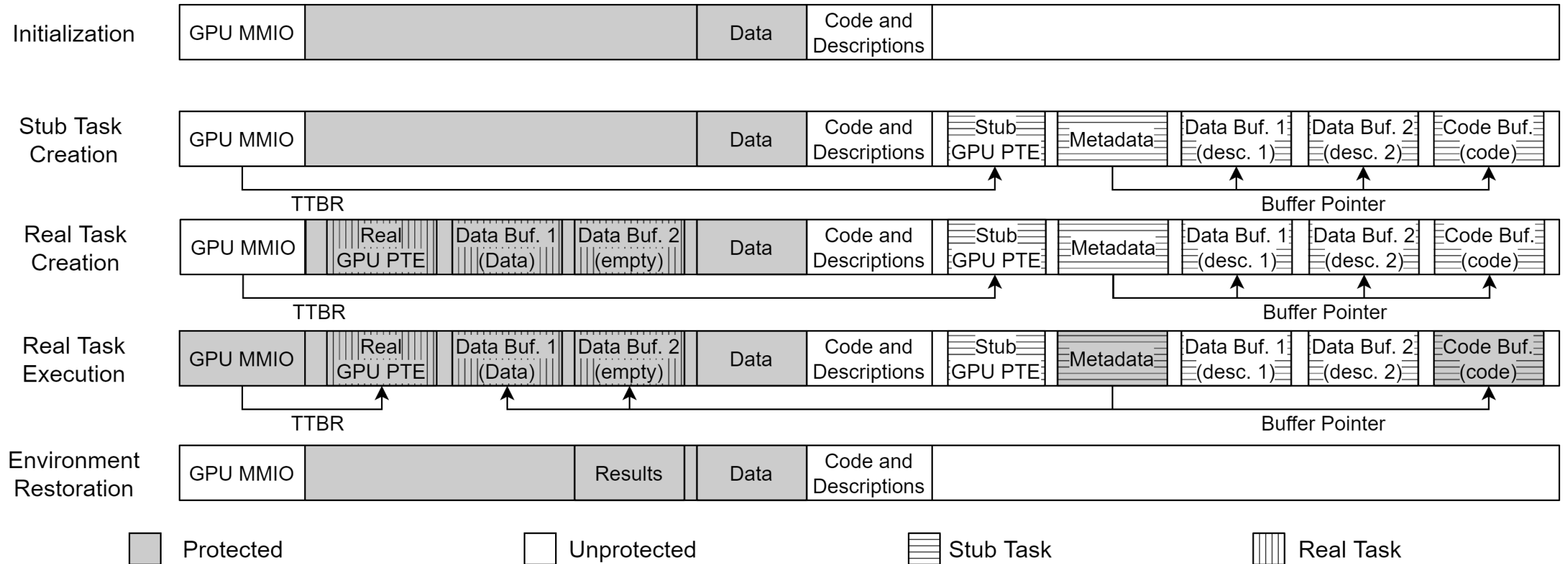


Goal 1: Compatibility with Arm CCA



- CCA's realm-style architecture
 - Realms are managed by Normal World software but invisible to them
 - Can we adapt it with GPU's workflow?
- Solution: Shadow task mechanism
 - Host schedules **stub tasks** for realms, with no sensitive data
 - When task submission, replace them with **real tasks**
 - Authentic data
 - Real data buffers created in realms
 - New GPU page table mappings

Goal 1: Compatibility with Arm CCA



Data buffer descriptions: critical information for creating real data buffers (e.g., buffer size, attributes, data to be filled, signatures)

Goal 2: Strong Data Security

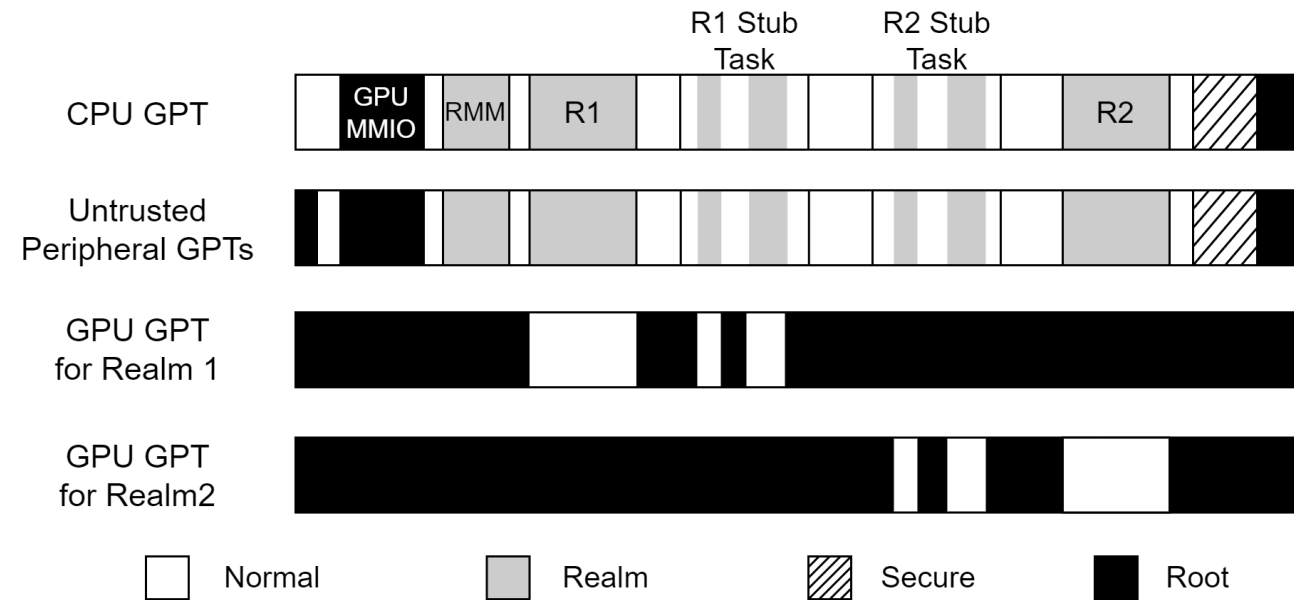


- RMM cannot directly manage GPUs
 - Unified-memory GPUs are regarded as **Normal** peripherals and cannot be re-configured as Realm peripherals
 - RMM cannot directly monitor **same-layer but untrusted** software (Normal/Secure hypervisors)
- Solution: Using **GPC on MMU/SMMU** to control memory access
 - Use a Granule Protection Table (GPT) to manage memory security view for CPU's MMU and peripherals' SMMU
 - Controlled by the highest-privilege Monitor

Goal 2: Strong Data Security



- Two Goals
 - Let **Normal** GPU access the protected regions
 - Two-way isolation between GPU environment and other components
- GPT for GPU:
 - Protected regions are **Normal** (accessible) state
 - Other regions are **inaccessible**
- GPT for CPU and untrusted peripherals:
 - Protected regions are **Realm** state
 - Protect GPU MMIO



Goal 2: Strong Data Security

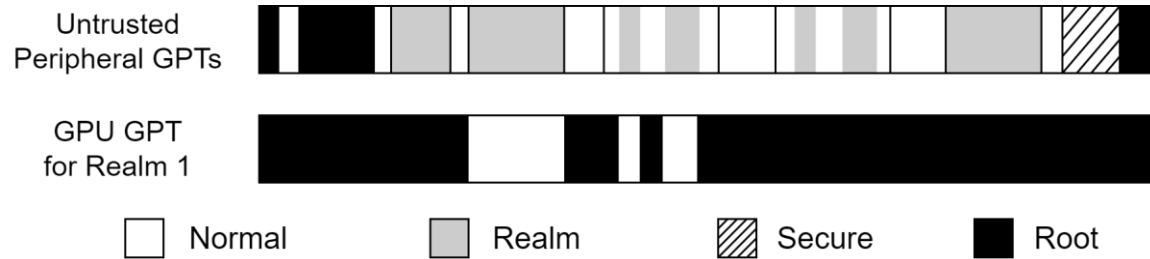


- Overall, we achieve two-way isolation for GPU computing
 - For GPU's SMMU GPC, switch to target GPU SMMU GPT
 - Synchronize the protection on CPU and Untrusted peripheral GPTs
- We also ensure GPU exclusivity for each real task
 - Protect GPU MMIO during the computing
 - Check GPU status (e.g., whether hiding malicious tasks) before real task submission
 - Clear GPU (e.g., TLB and cache) after real task computing

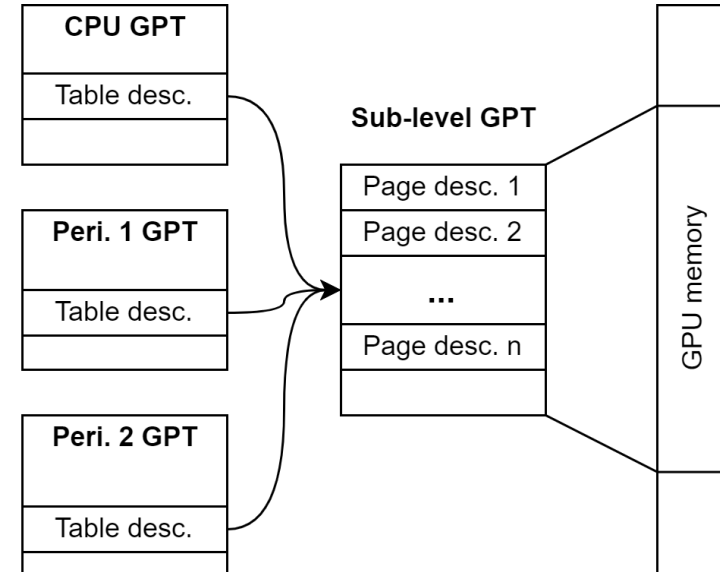
Goal 3: Low Performance Overhead



- Optimize GPT initialization and synchronization



Only store accessible (Normal) and non-accessible (Root) info in Realm's GPU SMMU GPT

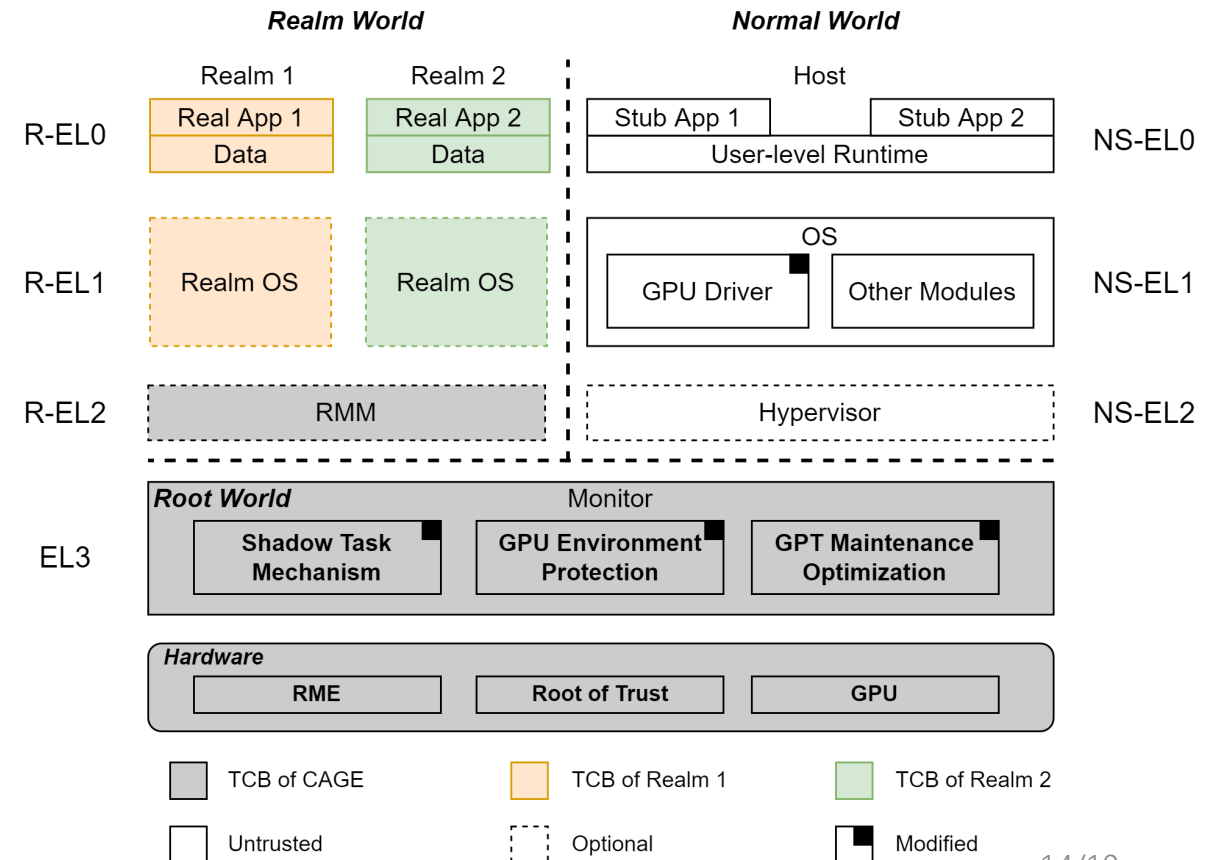


Use the same sub-level GPT to manage access from CPU and untrusted peripherals

Additional Goal: Hardware Compatibility



- Design and implement CAGE without hardware modification
 - Leverage current Realm Management Extensions (RME)
 - Generic unified-memory GPU



Functionality Prototype Implementation



- Environment
 - Arm FVP Base RevC-2xAEMvA, with RME enabled
- TCB:
 - ATF v2.8 (0.4M LoC) with 1.3K LoC additions
 - Realm isolation software (e.g., TF-RMM with 26K LoC)
- Not introduce GPU software stacks to Realms or CAGE's TCB

Security Evaluation



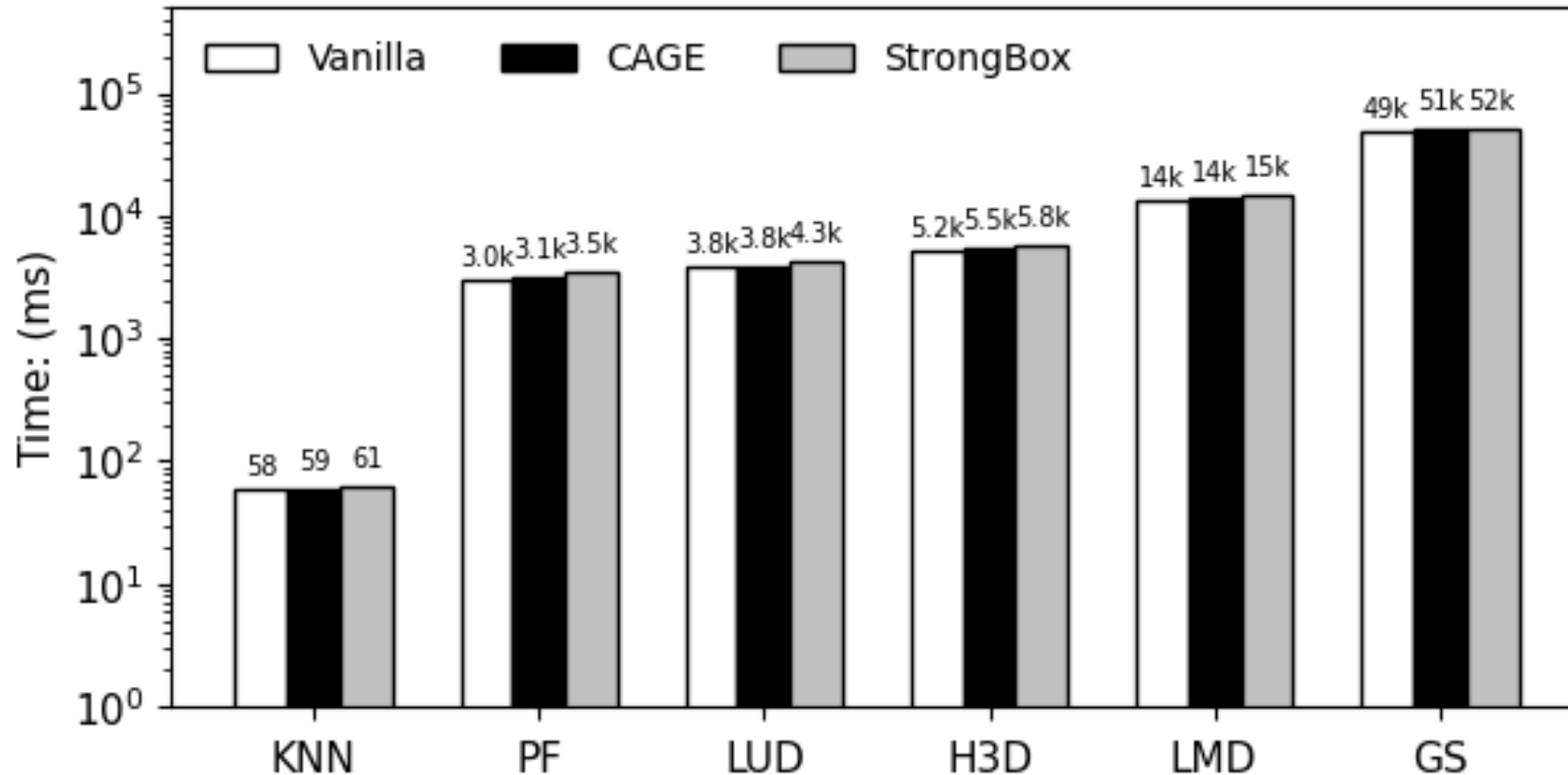
Adversary Type	Attack Scenarios	Defense
Untrusted software	Unauthorized memory access and modification	① ②
	Illegal GPU memory management	① ③
	Illegal GPU task scheduling	② ③
	Malicious GPU tasks	① ③
	Fake GPU and SMMU	④
	CPU GPC circumvention	① ⑤ ⑥
Peripherals	Malicious DMA	①
	Peripheral GPC circumvention	① ⑤ ⑥
Realms	Realm abuse	① ⑦

- ①: The GPC on CPU and peripheral access. ②: The integrity verification. ③: The Monitor checks.
④: The fixed MMIO address. ⑤: The hardware-assisted isolation of Root World.
⑥: The TLB invalidation. ⑦: The CPU-side memory isolation.

Evaluation



- Emulate CCA's security operations on Armv8 Juno R2 Board
 - Manage MMU/SMMU GPTs, read and write GPC registers ...
- Low (2.45%) performance overhead on the selected Rodinia benchmarks



Conclusions



- **CAGE** provides confidential GPU computing support for Arm CCA.
 - Follow Arm CCA's **realm-style architecture** to manage confidential GPU computing
 - Ensure **strong data security** with CCA's existing security hardware primitive
 - Adapt to Arm endpoints and servers with **low performance overhead** and **no hardware modification**
- Source code
 - <https://github.com/Compass-All/NDSS24-CAGE>



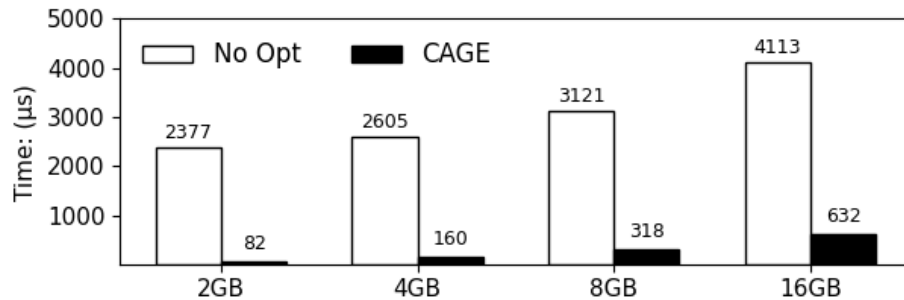
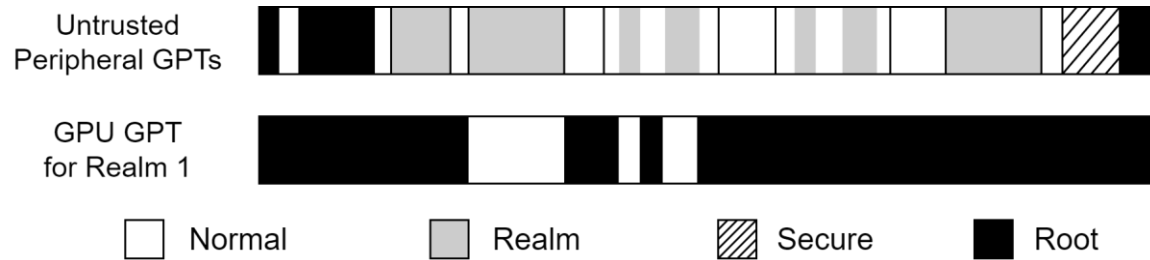
Thank You!

Project Repo: <https://compass.sustech.edu.cn/cage/>

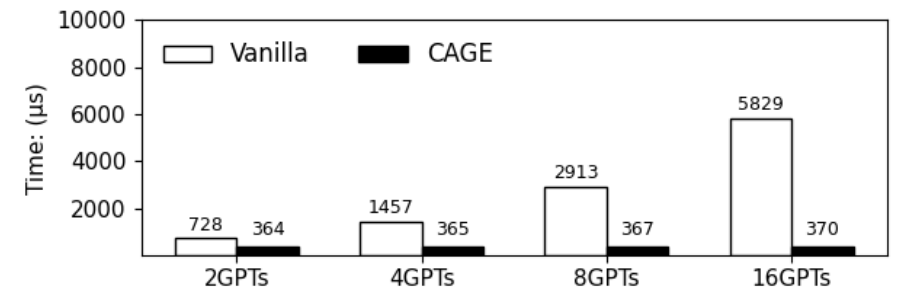
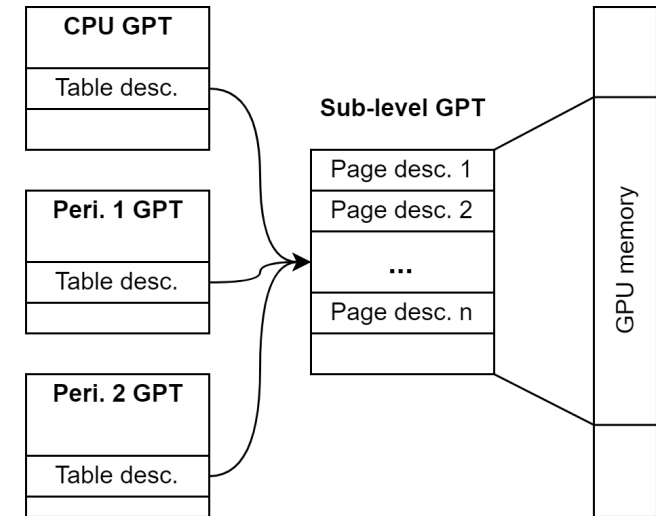
Backup Slides

Performance Evaluation

- Optimize GPT initialization and synchronization



Mitigate 84.63% – 96.55% performance overhead of GPU GPT initialization

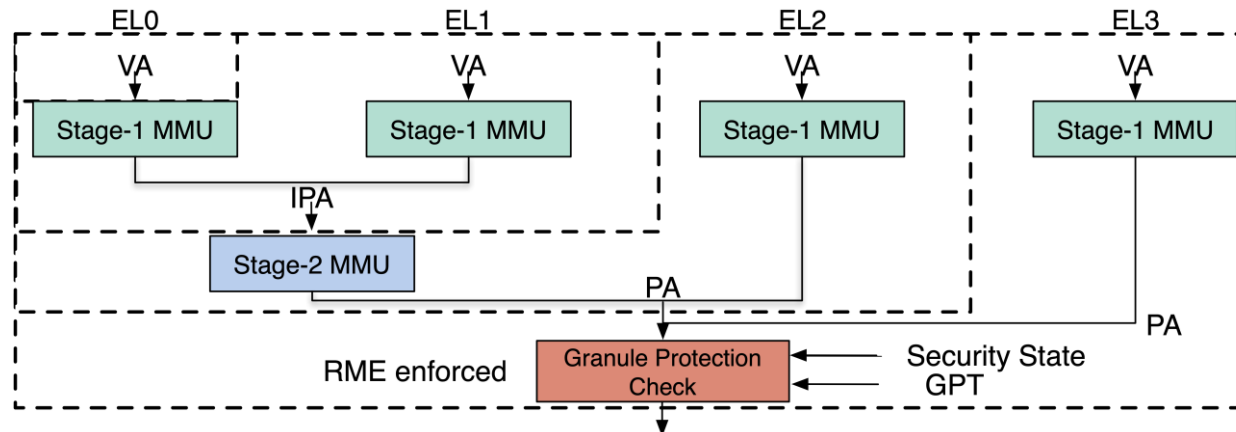


Mitigate 50.01% – 93.65% performance overhead on synchronizing multiple GPTs

Granule Protection Check (GPC)

- GPC can be enabled in CPU MMUs and peripheral SMMUs, indicating the security view of the connected CPU/peripheral.
- Such security view is managed by **Granule Protection Table (GPT)**
- Specifically, GPT specifies what physical address spaces (PAS) a memory page belongs to

Security state	Normal PAS	Secure PAS	Realm PAS	Root PAS
Normal	✓	×	×	×
Secure	✓	✓	×	×
Realm	✓	×	✓	×
Root	✓	✓	✓	✓



Comparison to state-of-the-art

	GPU Memory	Protection Mechanism	Minimal TEE TCB	Sharing of GPU	HW Compatibility	Native Size of Trusted Components (LoC)	Runtime TCB Increment (LoC)
Non-Arm Designs							
Graviton [76]	Dedicated	Customized GPU HW	✓	✓	✗	Firmware (N/A)	N/A
HIX [52]	Dedicated	Intel SGX	✗	✓	✗	Firmware (N/A)	N/A
HETEE [81]	Dedicated	Customized HW	✗	✗	✗	Firmware (N/A)	N/A
CURE [31]	Unified	Customized HW	✗	✗	✗	Firmware (N/A)	3.1K on Firmware
HoneyComb [61]	Dedicated	AMD SEV-SNP	✓	✓	✓	Security Monitors (N/A)	83K on Security Monitors
Arm Designs							
Cronus [54]	Dedicated	TZASC + S-Hyp	✗	✓	✓	S-Hyp (34K) + Monitor (0.4M)	6.4K on S-Hyp
StrongBox [39]	Unified	TZASC + S2 trans.	✓	✗	✓	OP-TEE (0.3M) + Monitor (0.4M)	1.2K on Monitor
GR-T [70]	Unified	TZASC + Remote VM	✗	✗	✓	VM (17M) + OP-TEE (0.3M) + Monitor (0.4M)	1K on VM + 3.5K on OP-TEE
ACAI [73]	Unified / Dedicated	GPC + RMM	✗	✗	✓	RMM (26K) + Monitor (0.4M)	0.4K on RMM + 1.6K on Monitor
CAGE	Unified	GPC	✓	✓	✓	CPU-side Isolation (2–26K) + Monitor (0.4M)	1.3K on Monitor