Dissecting Performance Overheads of *Confidential Computing in GPU-based Systems*

Yang Yang, Mohammad Sonji, Adwait Jog

{yangyang, npv2tk, ajog} @virginia.edu

Insight Computer Architecture Lab

UNIVERSITY of VIRGINIA

Executive Summary

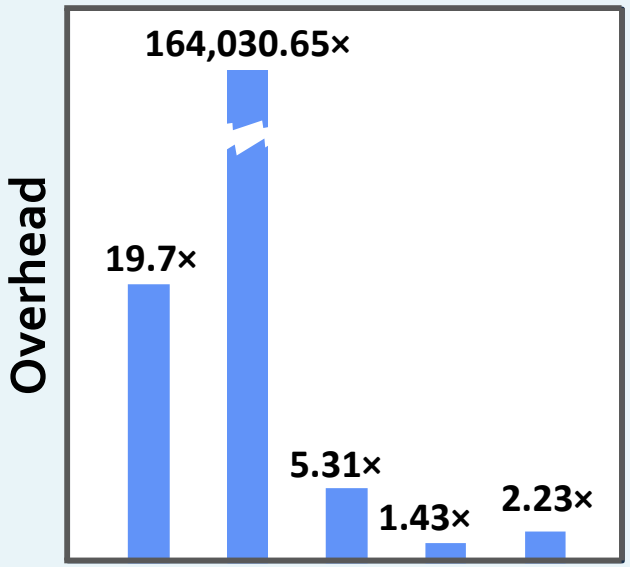
Problem:

- Confidential Computing **with GPU** is emerging
- Existing work lacks **detailed** performance **dissection**
- **Optimization insights** remain unclear due to limited characterization

Goal: Dissect the **performance overheads** of GPU-based confidential computing that can motivate various optimizations to address them.

Our results on real hardware show that GPU-based CC comes with:

- **Data Movement** Overhead (up to 19.7×)
- Low **Encryption** Throughput (3.36 GB/s)
- Kernel **Execution** Time (up to 164,030.65× with **UVM**)
- Kernel **Launch** Overhead (up to 5.31×)
- **Queueing** (1.43× and 2.23× on average)
- **Fusion**, **overlapping** and **quantization**

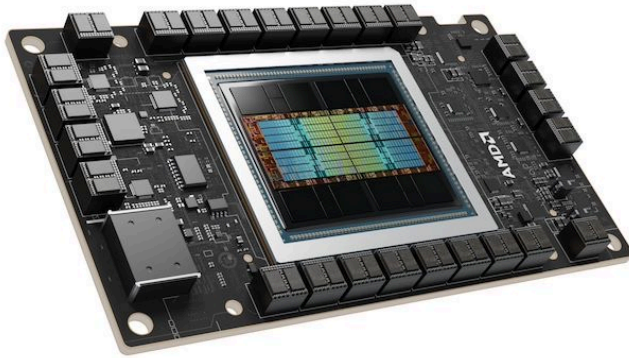


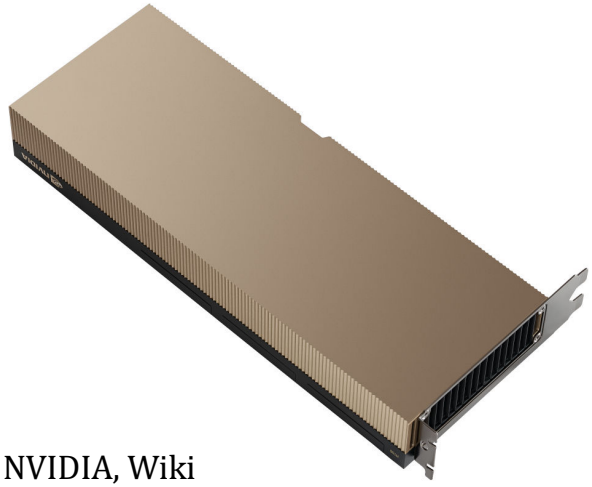
https://github.com/insight-cal-uva/hcc-ispass25-artifact

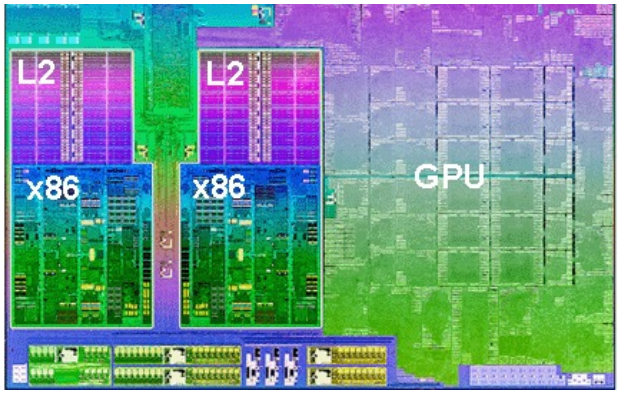
Graphics Processing Units (GPUs)

DL/ML/AI | Graphics | HPC | Crypto | Graphs | Genomics

Abundant Data and Task-level Parallelism







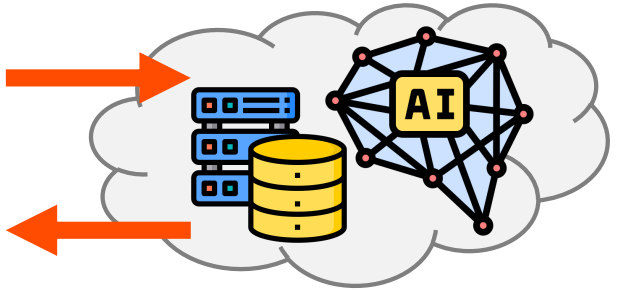
Credit/Source: AMD, NVIDIA, Wiki

High Throughput + Energy Efficiency

Needs for Security and Privacy

- ***Data is Private***
  - Client privacy (chat bot, …)
  - Regulated (GDPR, PII, …)

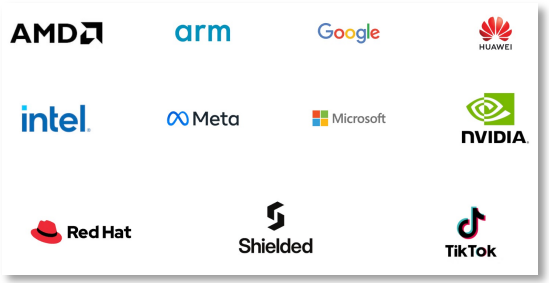
GDPR.EU

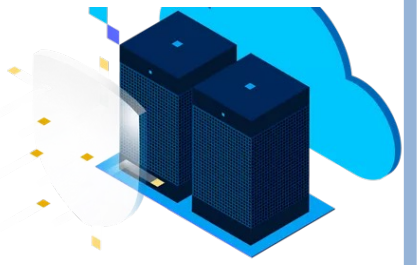


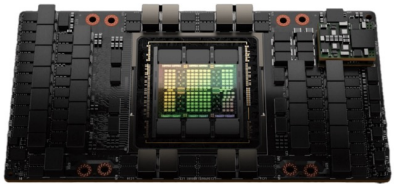
- ***Data is Massive***
  - Outsource compute & data to public cloud Infra.

Can cloud service providers be trusted?

Trusted Computing





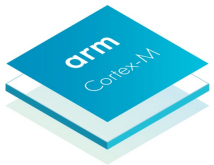


The first commercial GPU TEE

NVIDIA HCC **2023**

CPUs With TDX

Intel SPR **2023** Intel EMR **2023**



CONFIDENTIAL COMPUTING CONSORTIUM

ARM TrustZone **2002**

Intel SGX **2015**

AMD SEV **2016**

CC Consortium **2019**

Intel TDX **2020**

AMD SEV-SNP **2020**

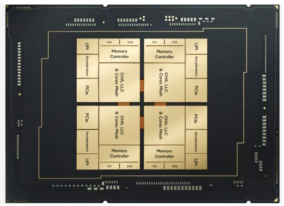
ARM CCA **2021**

VM-based Isolation

RISC-V KeyStone **2019**

RISC-V Penglai **2020**

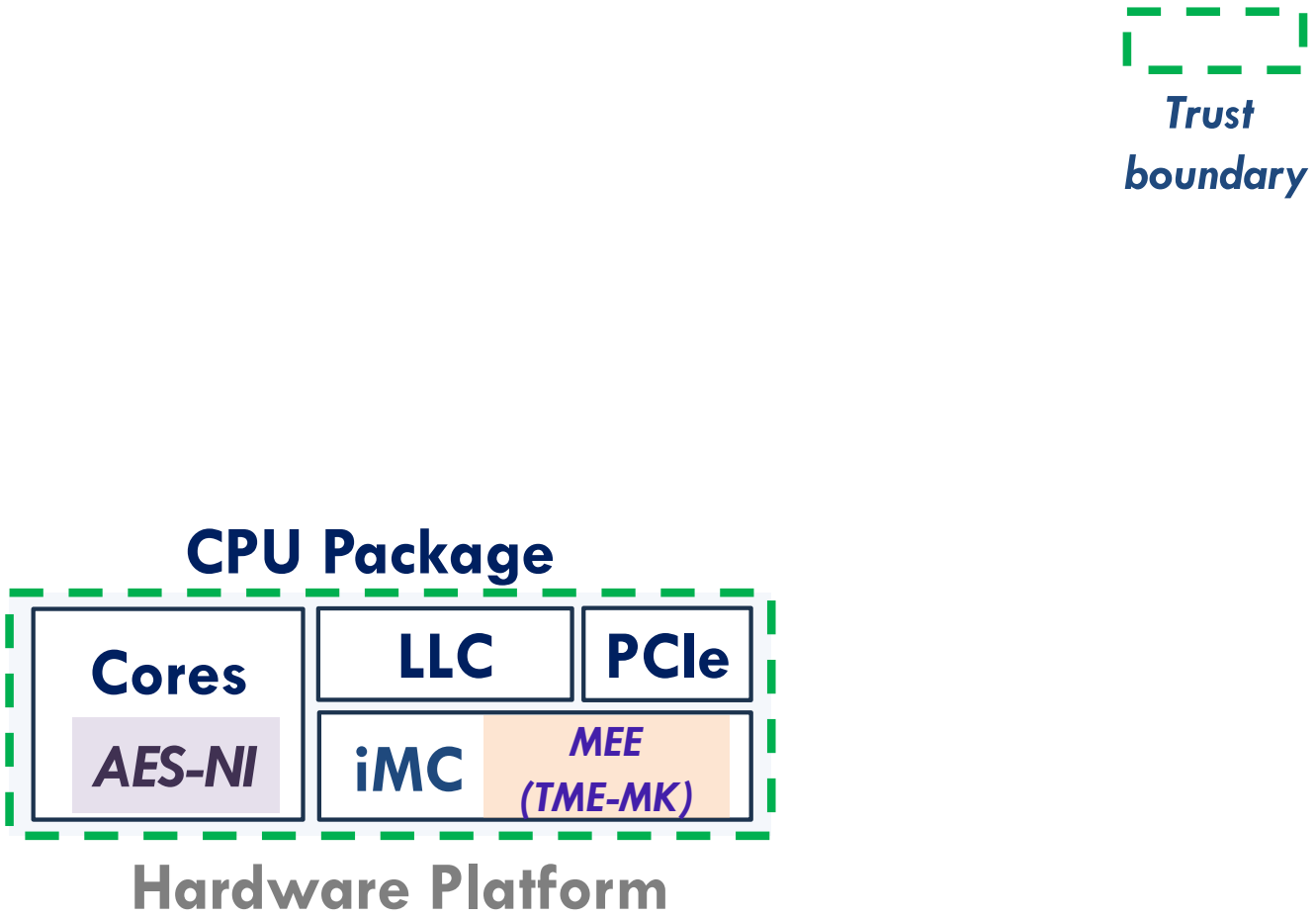
RISC-V®



GPU-based CC

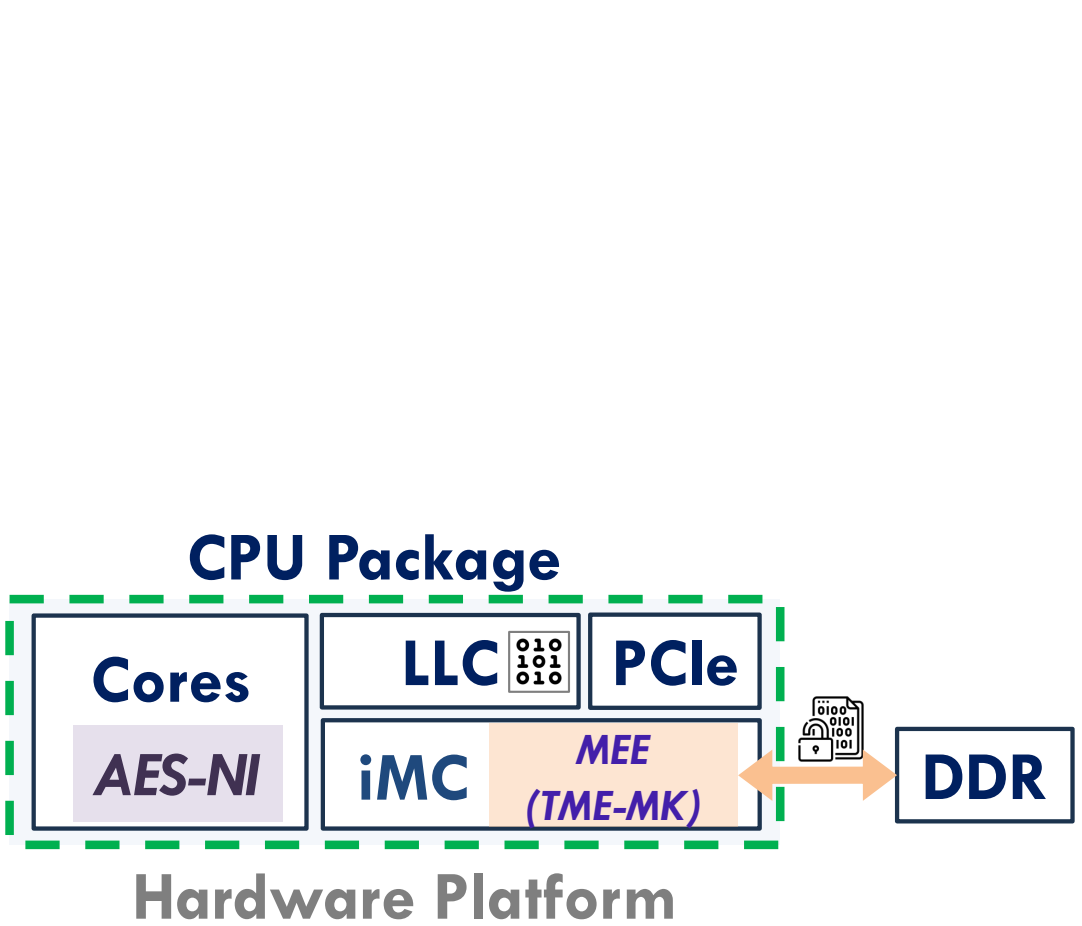
Credit/Source: ARM/INTEL/AMD/CCC/RISC-V/NVIDIA/Wiki

Confidential Computing



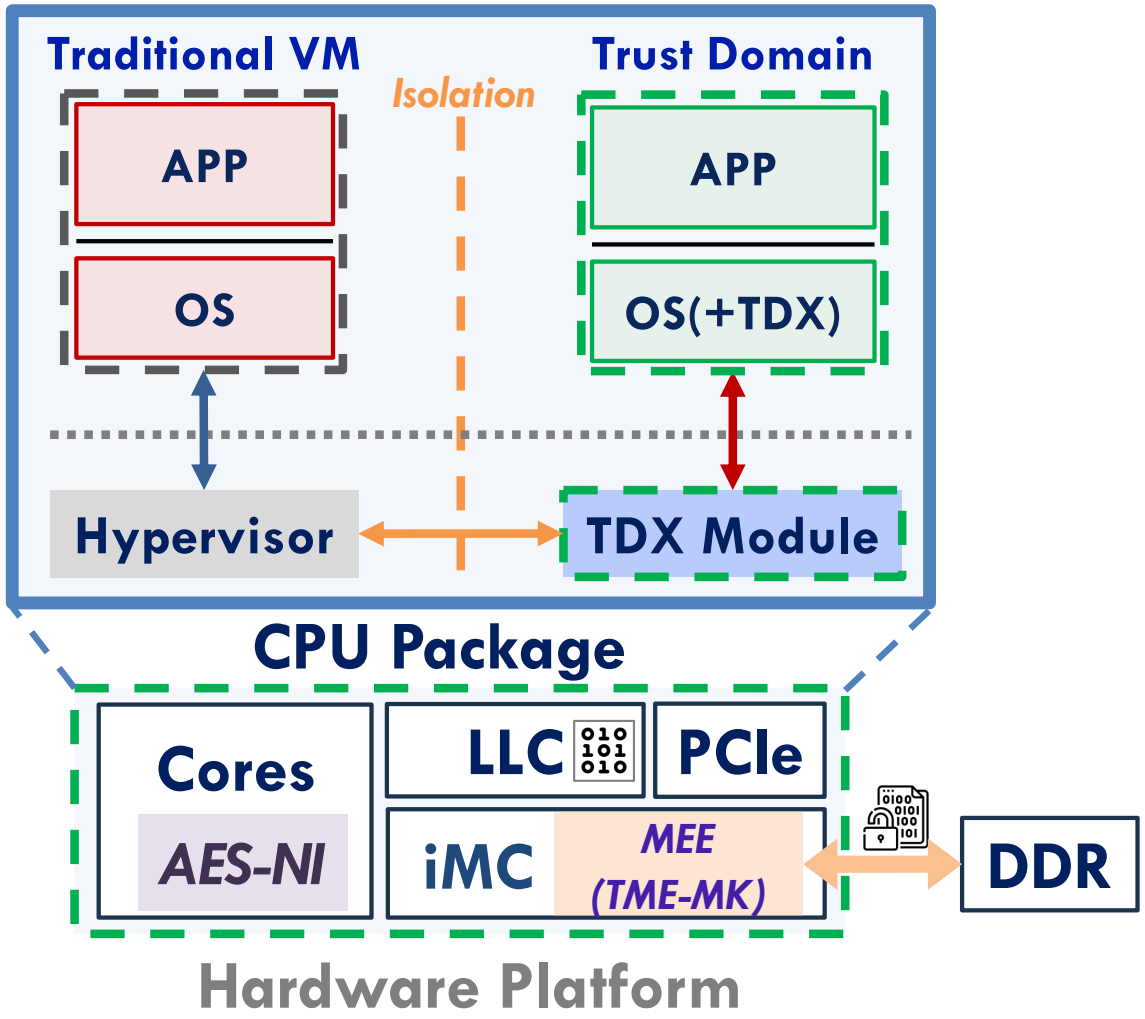
MEE: Memory Encryption Engine

Confidential Computing



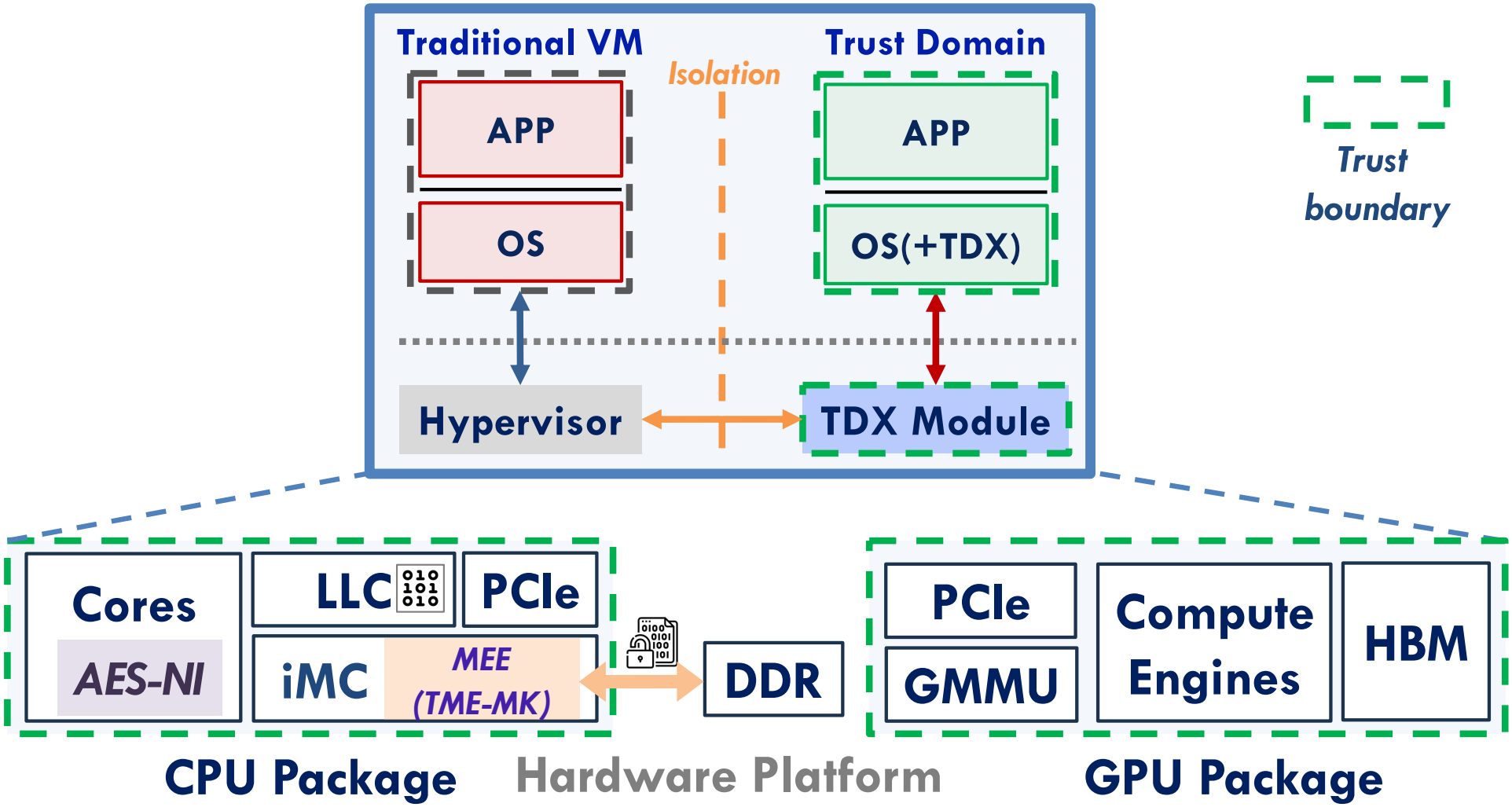
MEE: Memory Encryption Engine

Confidential Computing



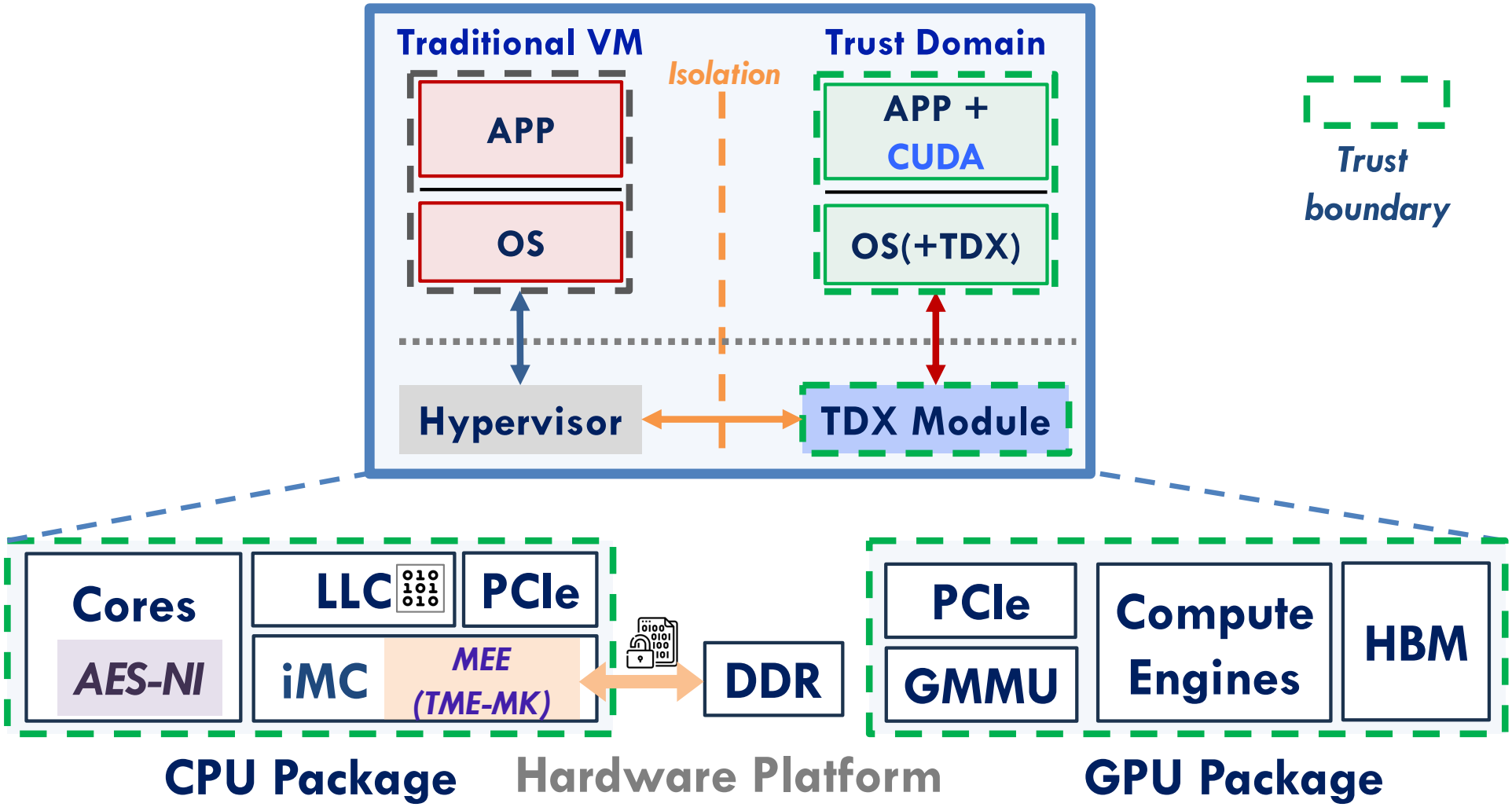
MEE: Memory Encryption Engine

Confidential Computing with GPU



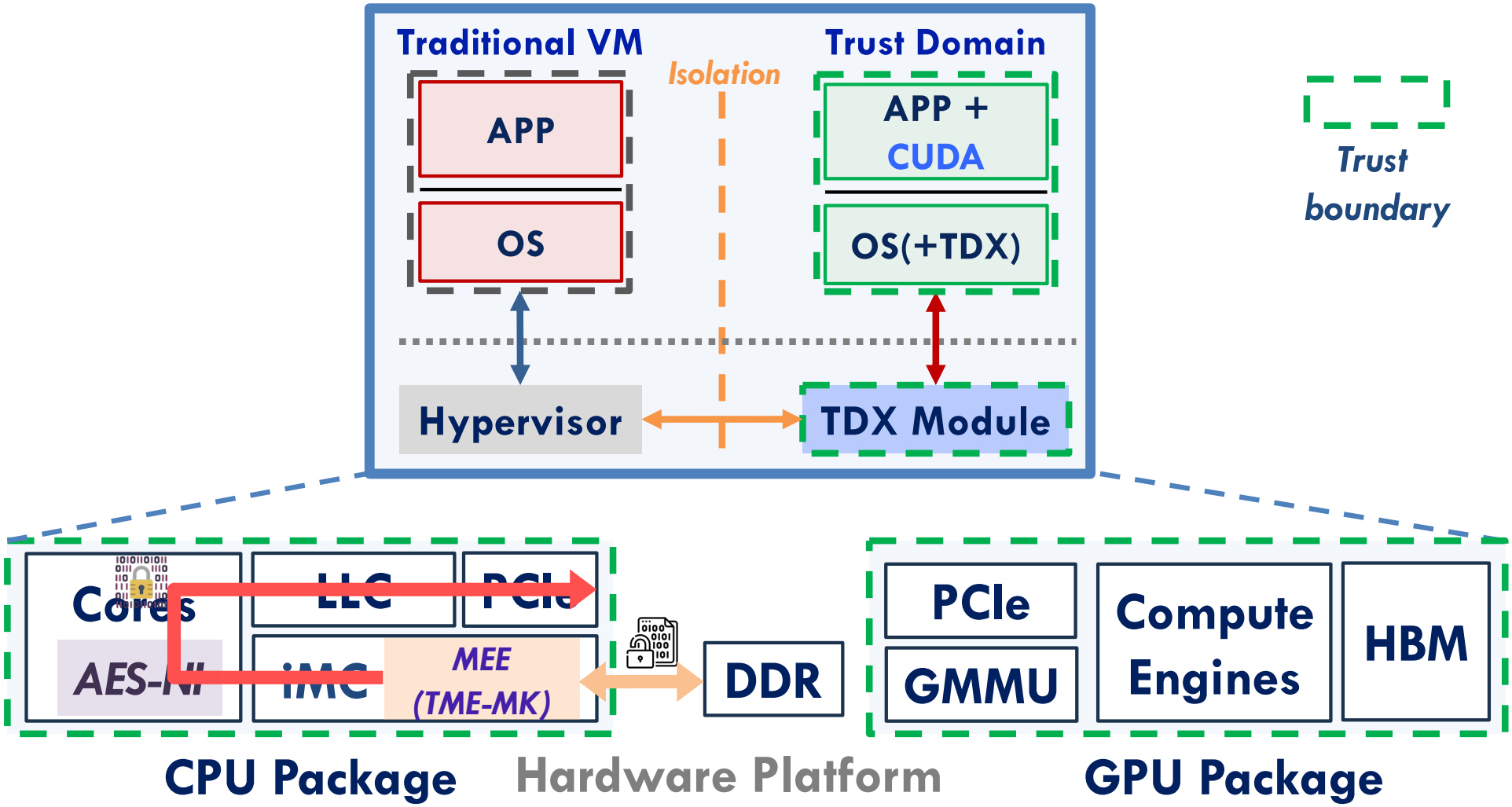
MEE: Memory Encryption Engine

Confidential Computing with GPU



MEE: Memory Encryption Engine

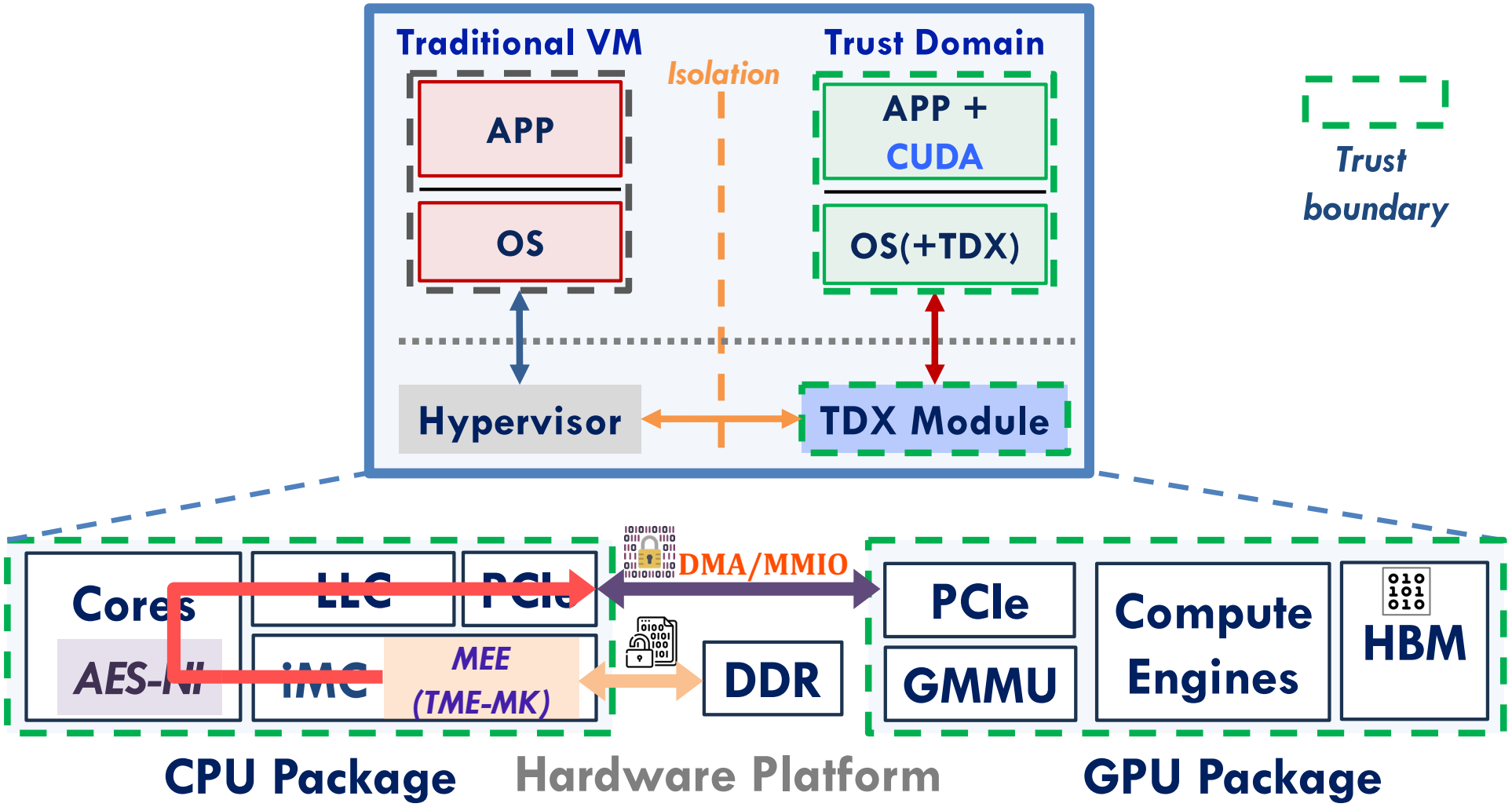
Confidential Computing with GPU



MEE: Memory Encryption Engine

Confidential Computing with GPU

Communication (MMIO/DMA) will trigger #VE, TD will context-switch



MEE: Memory Encryption Engine

Real World Setup

https://github.com/insight-cal-uva/hcc-ispass25-artifact

System Configuration

Configuration	Details
CPU	2× 5th Gen Intel Xeon 6530 Gold @2.1GHz, 32 cores
TME-MK	Auto bypass enabled
OS	Ubuntu 22.04.5 LTS (Linux 6.2.0, tdx patched)
Hypervisor	QEMU 7.2.0 (tdx patched)
TDX Tools	TDX 1.5 (tag 2023ww15)
GPU	NVIDIA H100 NVL, 94GB HBM3, PCIe 5.0 ×16 CUDA 12.4, Driver 550.127.05

CC-capable CPUs: support TDX or SEV-SNP

H100 is the first GPU that supports CC.

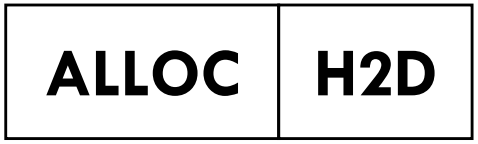
Problem:

- Confidential Computing **with GPU** is emerging
- Existing work lacks **detailed** performance **dissection**
- **Optimization insights** remain unclear due to limited characterization

Performance Model

ALLOC

Performance Model

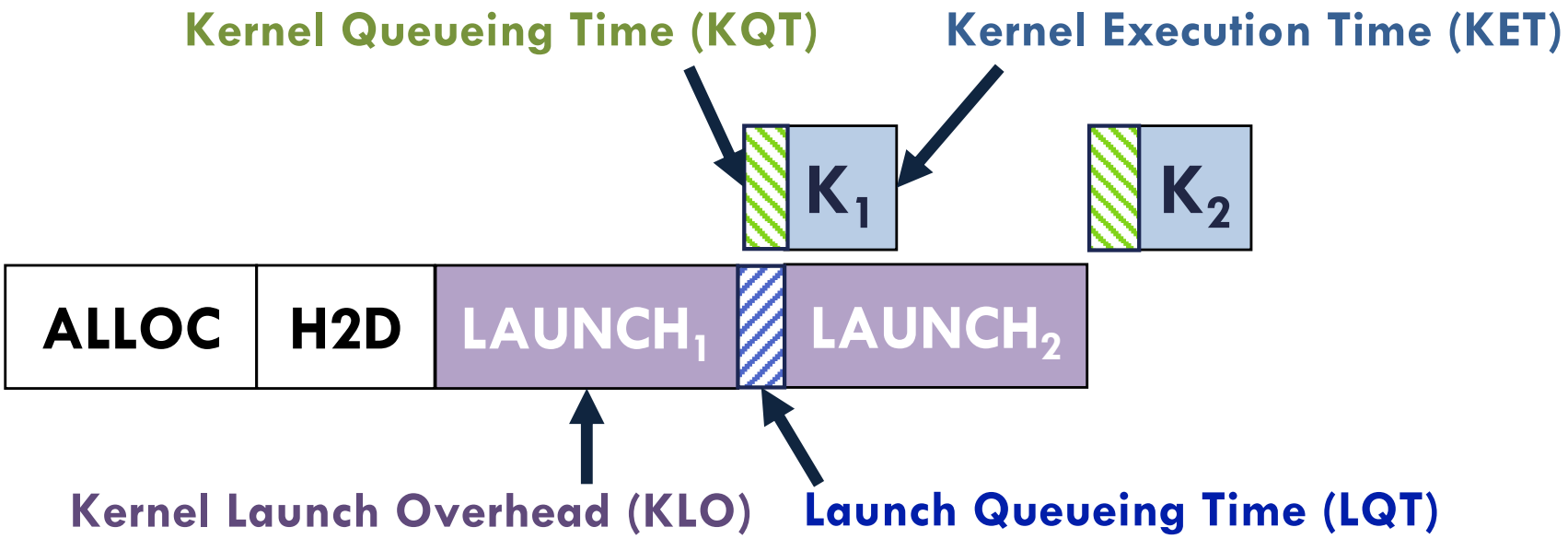


H2D: host (CPU) to device (GPU) copy

Performance Model

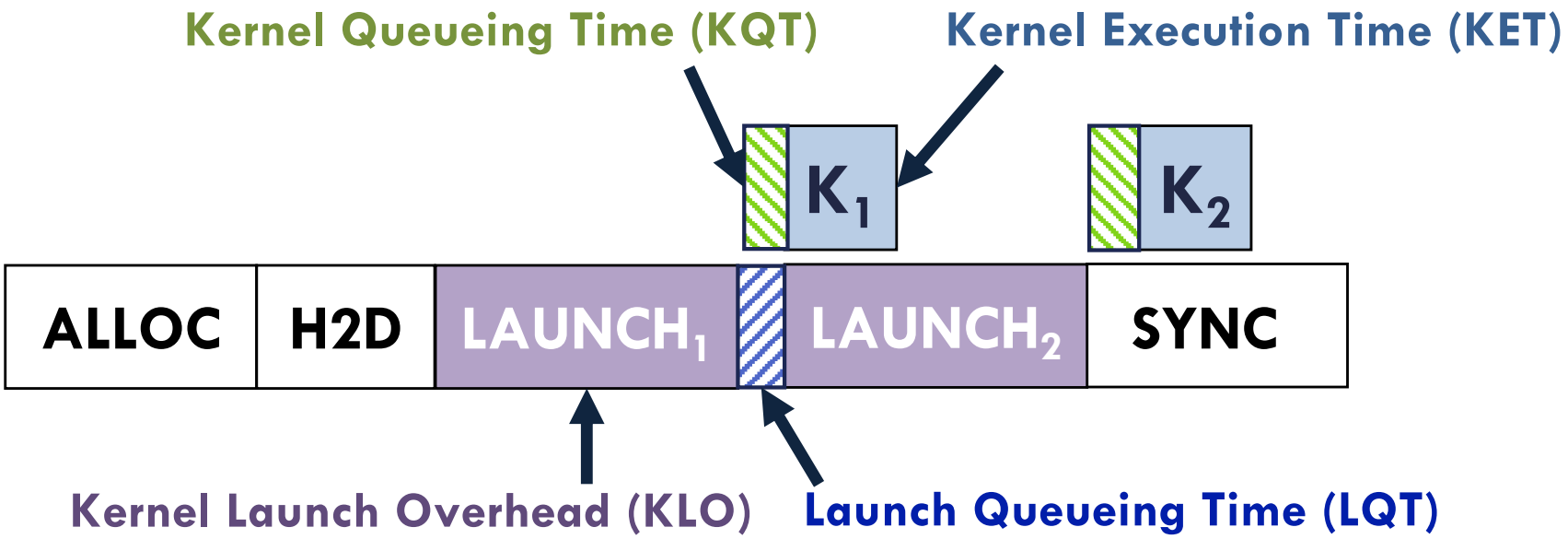
H2D: host (CPU) to device (GPU) copy

Performance Model



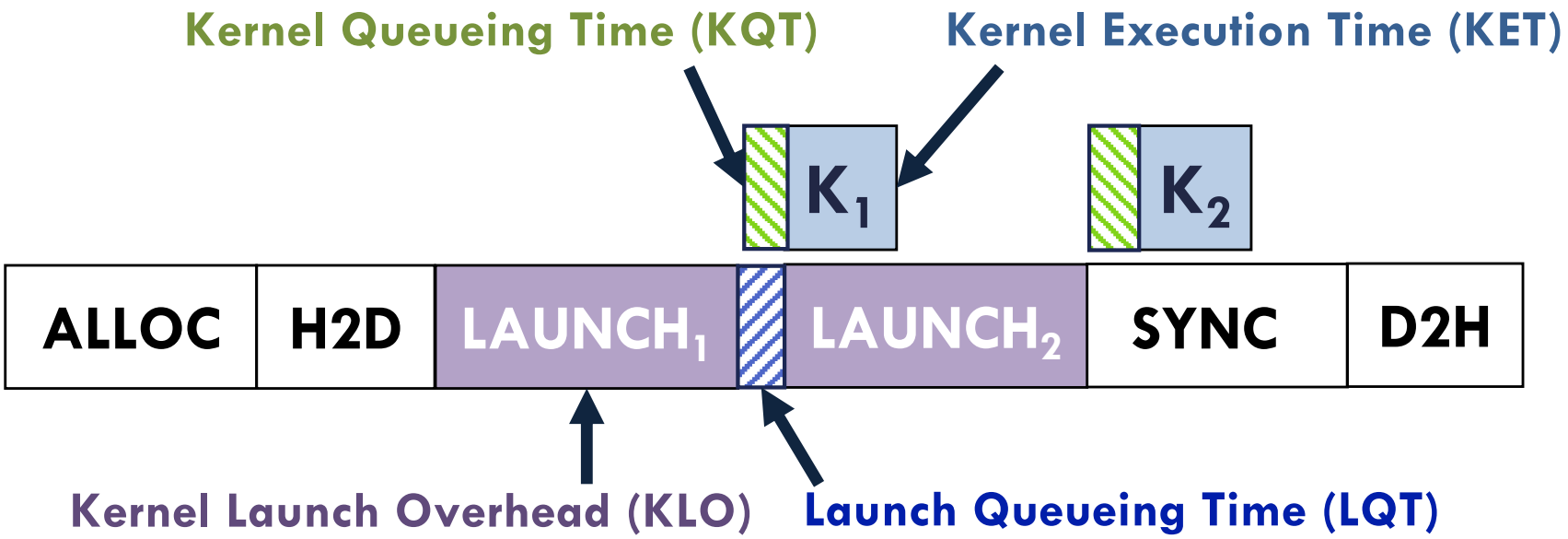
H2D: host (CPU) to device (GPU) copy

Performance Model



H2D: host (CPU) to device (GPU) copy

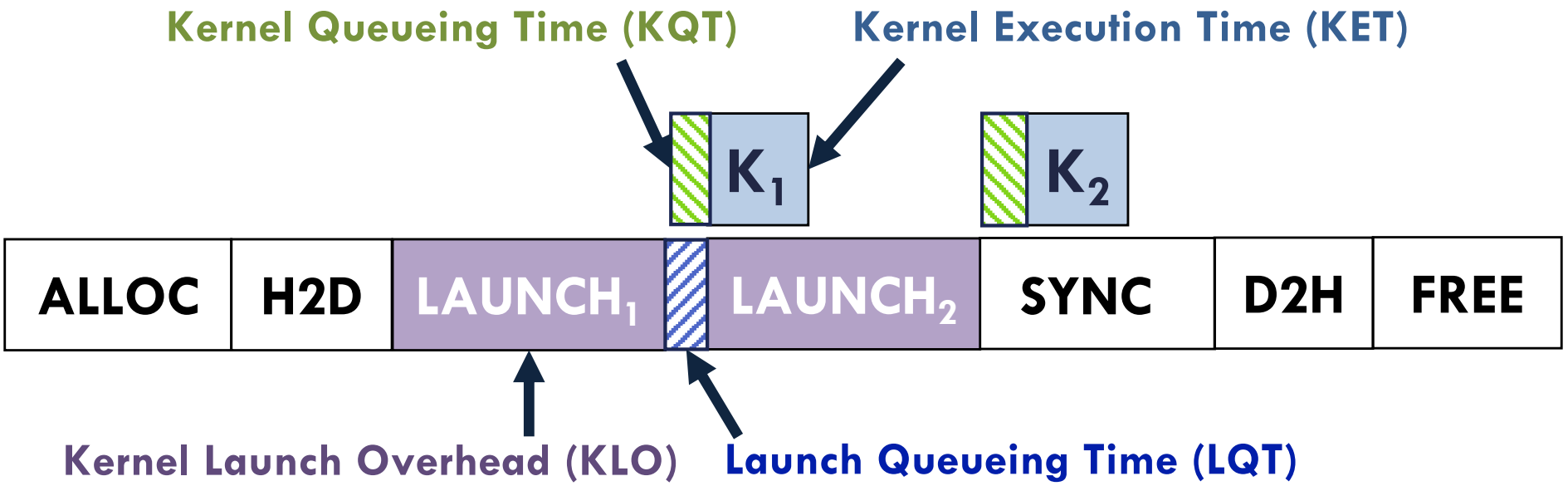
Performance Model



D2H: device (GPU) to host (CPU) copy

H2D: host (CPU) to device (GPU) copy

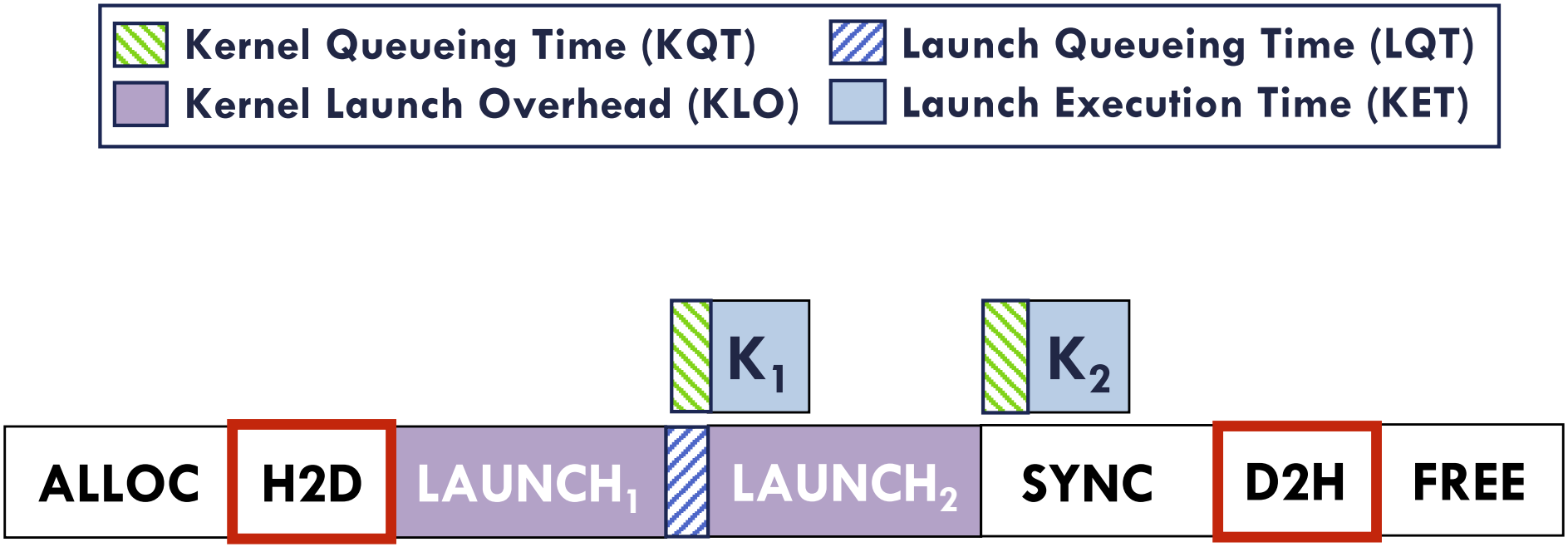
Performance Model



D2H: device (GPU) to host (CPU) copy

H2D: host (CPU) to device (GPU) copy

Performance Model



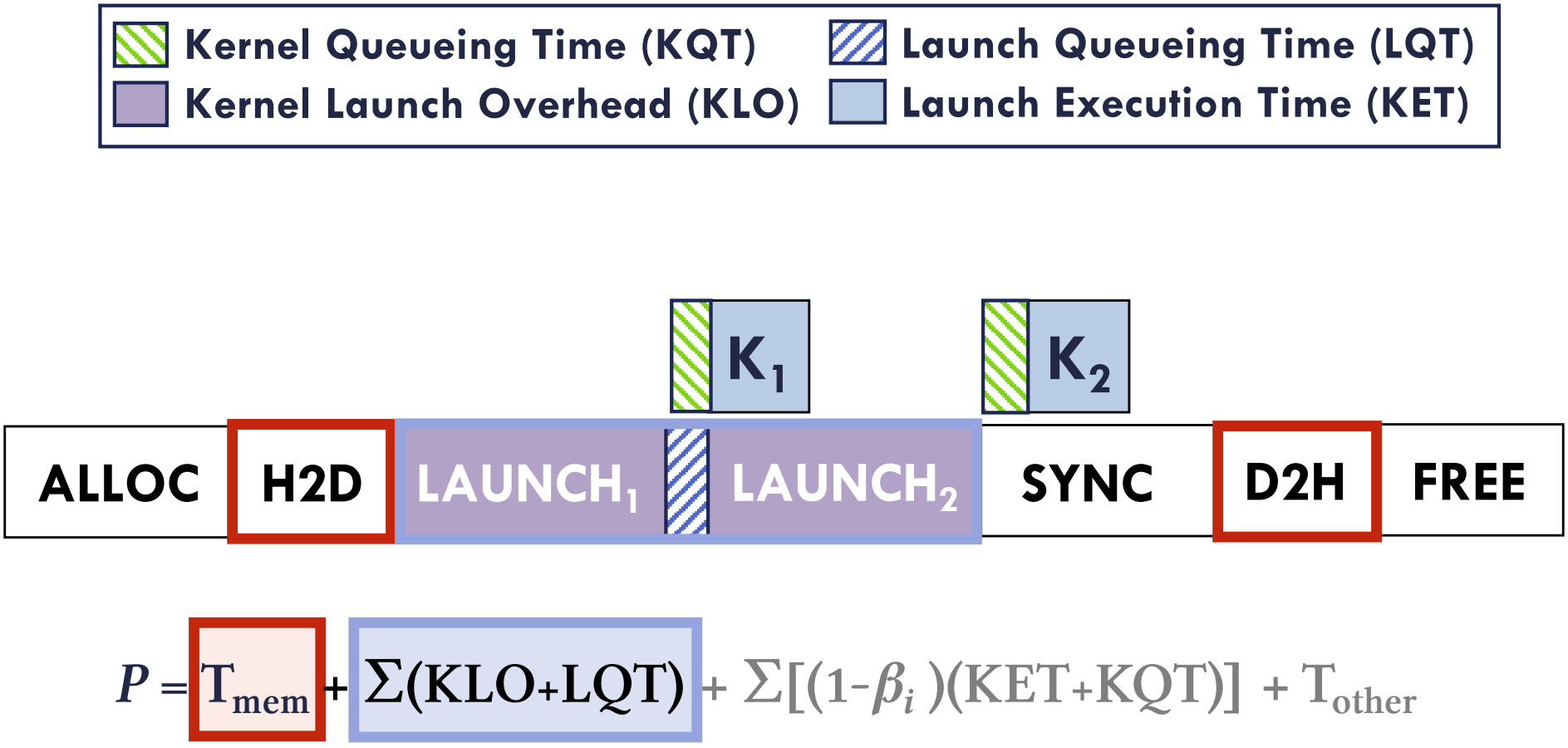
$$P = T_{mem} + \Sigma(KLO+LQT) + \Sigma[(1-\beta_i)(KET+KQT)] + T_{other}$$

Time on memory copy

D2H: device (GPU) to host (CPU) copy

H2D: host (CPU) to device (GPU) copy

Performance Model

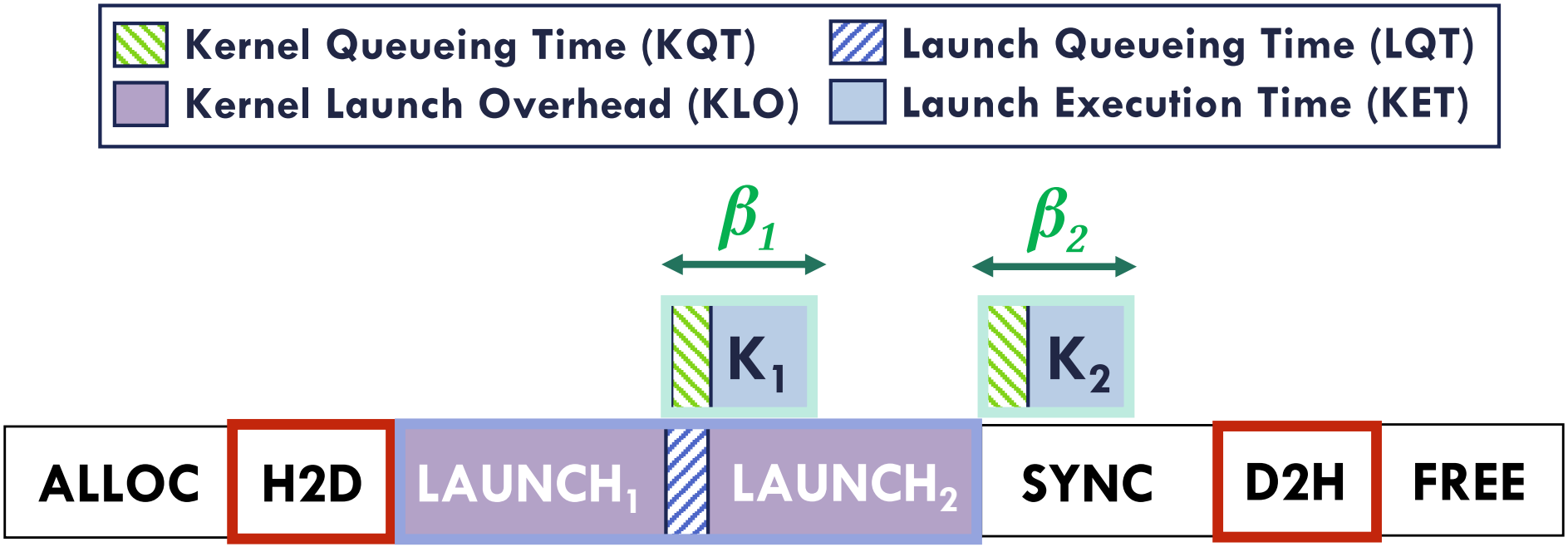


$$P = T_{mem} + \Sigma(KLO+LQT) + \Sigma[(1-\beta_i)(KET+KQT)] + T_{other}$$

D2H: device (GPU) to host (CPU) copy

H2D: host (CPU) to device (GPU) copy

Performance Model

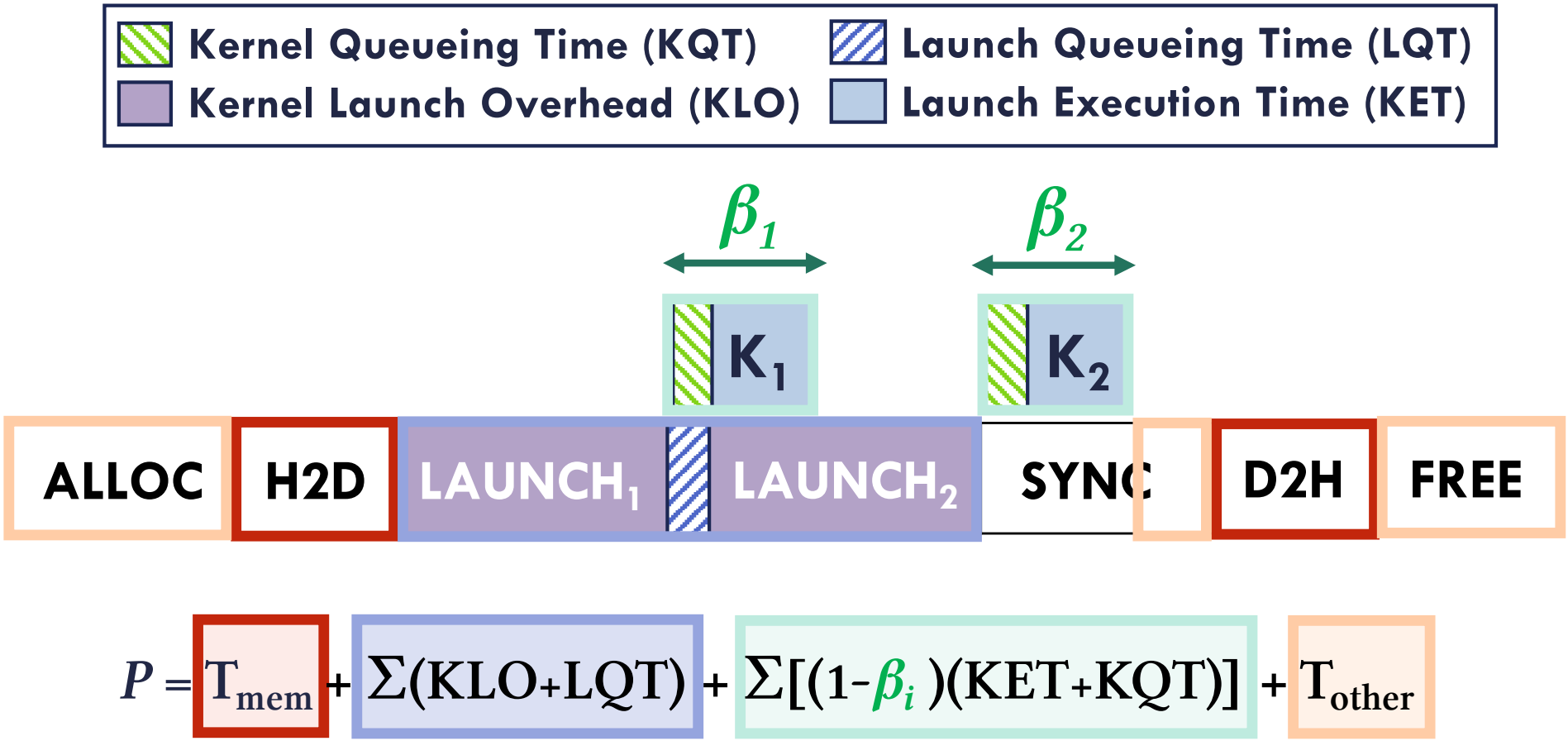


$$P = T_{mem} + \Sigma(KLO+LQT) + \Sigma[(1-\beta_i)(KET+KQT)] + T_{other}$$

D2H: device (GPU) to host (CPU) copy

H2D: host (CPU) to device (GPU) copy

Performance Model



$$P = T_{mem} + \Sigma(KLO+LQT) + \Sigma[(1-\beta_i)(KET+KQT)] + T_{other}$$

D2H: device (GPU) to host (CPU) copy

H2D: host (CPU) to device (GPU) copy

Performance Model

$$P = T_{mem} + \Sigma(KLO+LQT) + \Sigma[(1-\beta_i)(KET+KQT)] + T_{other}$$

D2H: device (GPU) to host (CPU) copy

H2D: host (CPU) to device (GPU) copy

Performance Model

$$P = T_{mem} + \Sigma(KLO+LQT) + \Sigma[(1-\beta_i)(KET+KQT)] + T_{other}$$

Focus of this presentation

- Data movement (H2D, D2H, page migration)
- Encryption
- Kernel Execution
- Kernel Launch
- Queuing

Transfer Bandwidth

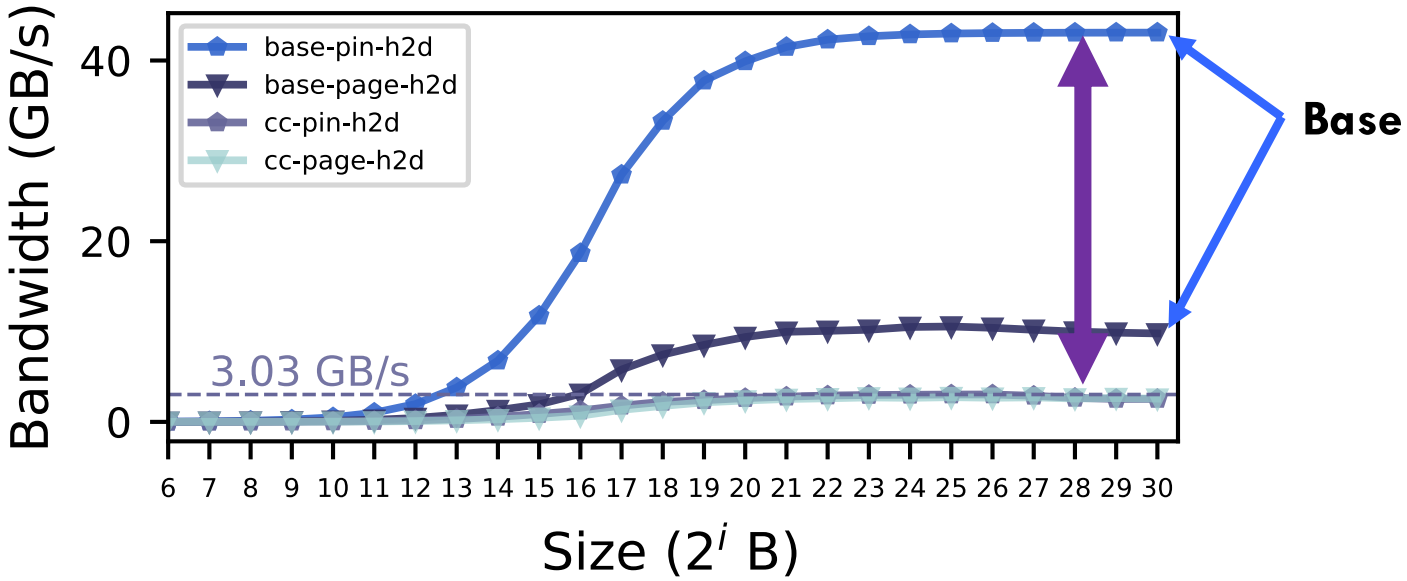


Fig.1 Copy bandwidth

Observation 1.

✓ PCIe bandwidth utilization in CC mode **drops significantly** compared to non-CC.

Transfer Bandwidth

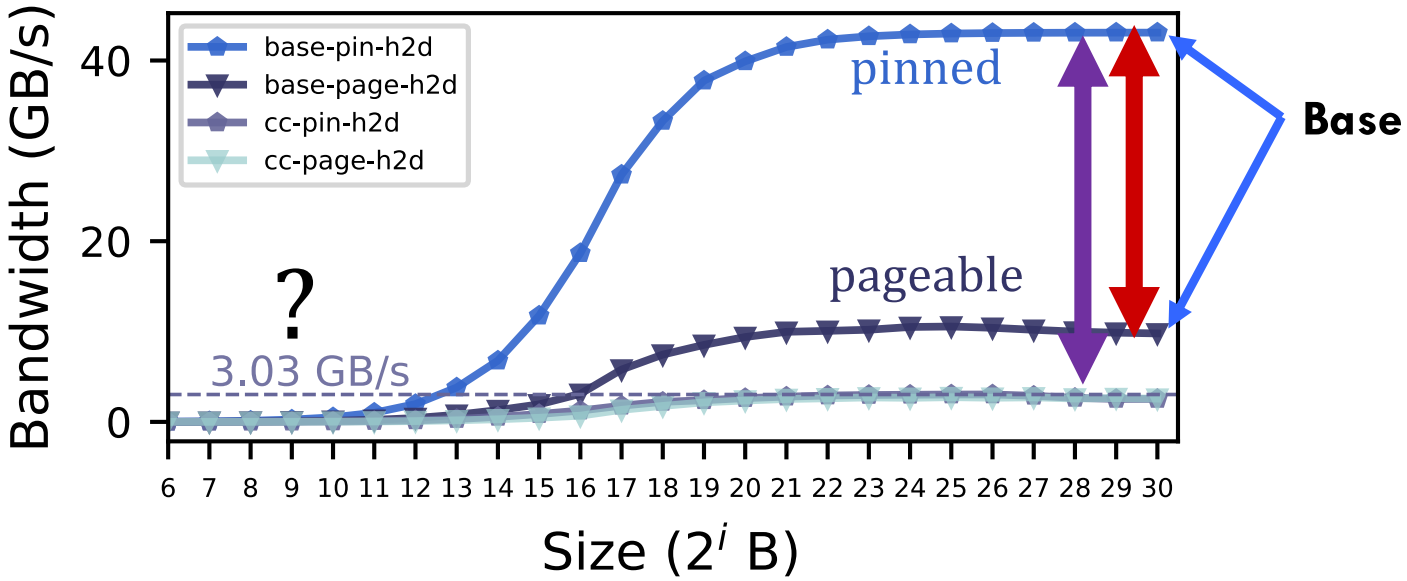


Fig.1 Copy bandwidth

Observation 1.

- ✓ PCIe bandwidth utilization in CC mode **drops significantly** compared to non-CC.
- ✓ Bandwidth **gap** between pageable and pinned memory observed in non-CC mode disappears in CC mode, suggesting that pinned memory relies on pageable mechanisms in CC mode.

Crypto Throughput

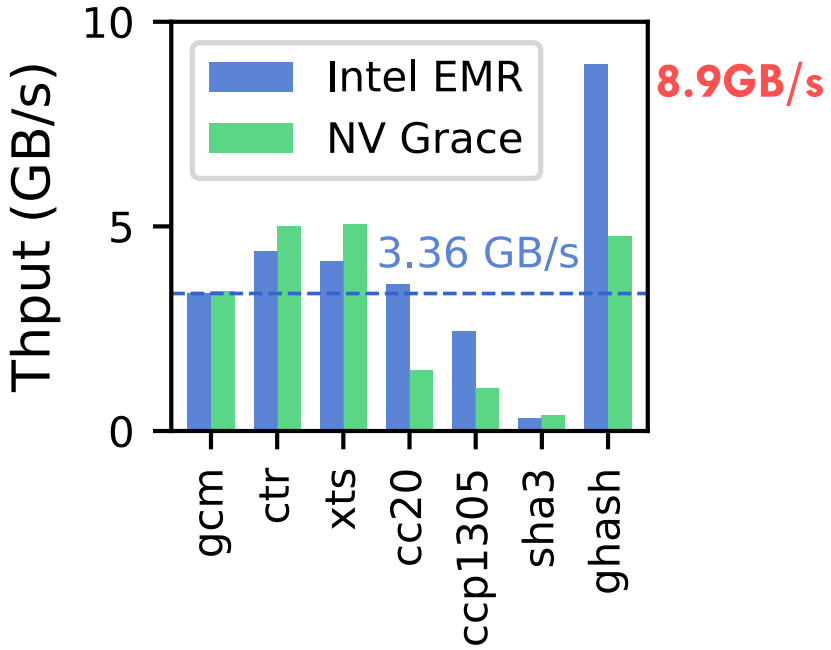


Fig.2 Crypto throughput

Observation 2.

- ✓ The absence of **dedicated hardware AES engines** results in low encryption throughput, even when using **AES-NI** acceleration.
- ✓ While *alternative cryptographic algorithms* may offer higher throughput, they often come at the cost of weaker security guarantees.

AES is on the critical path.

Kernel Execution Time

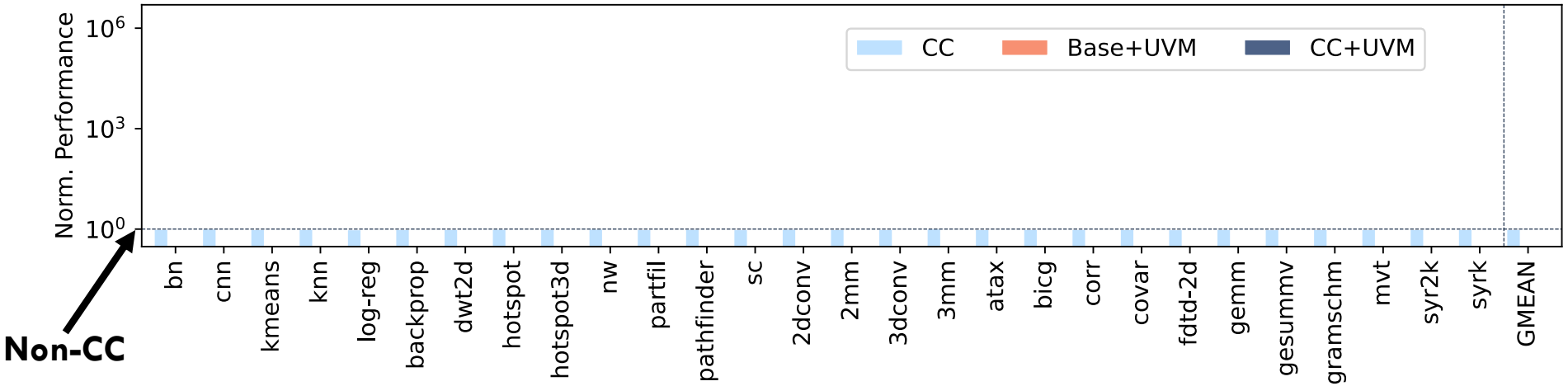


Fig.3 Normalized kernel execution time.

Observation 3.

- CC has **minimal** impact on **non-UVM** kernels (**0.48% increase**).

? **Execution is locked inside GPU, no interaction with CPU**

Kernel Execution Time

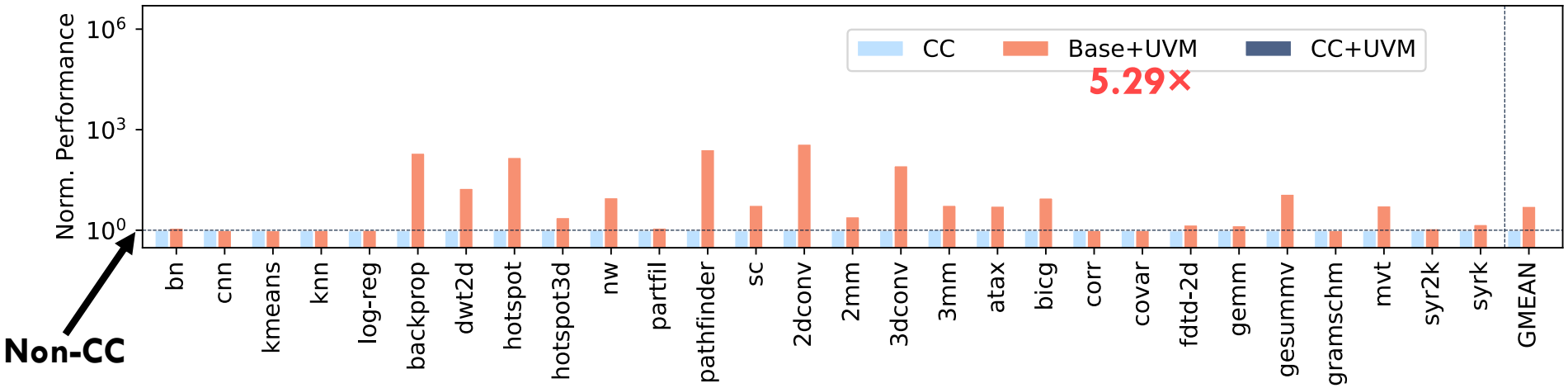
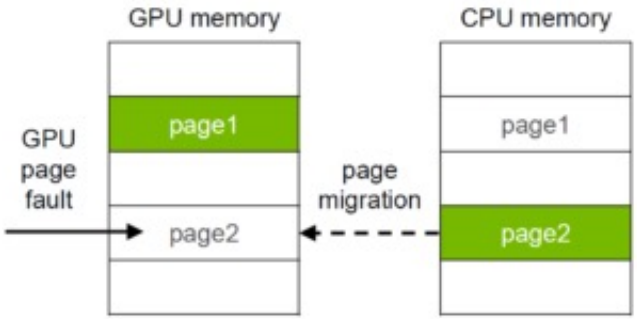


Fig.3 Normalized kernel execution time.

Single Address Space



Source: NVIDIA

Frequent CPU-GPU interaction!

Kernel Execution Time

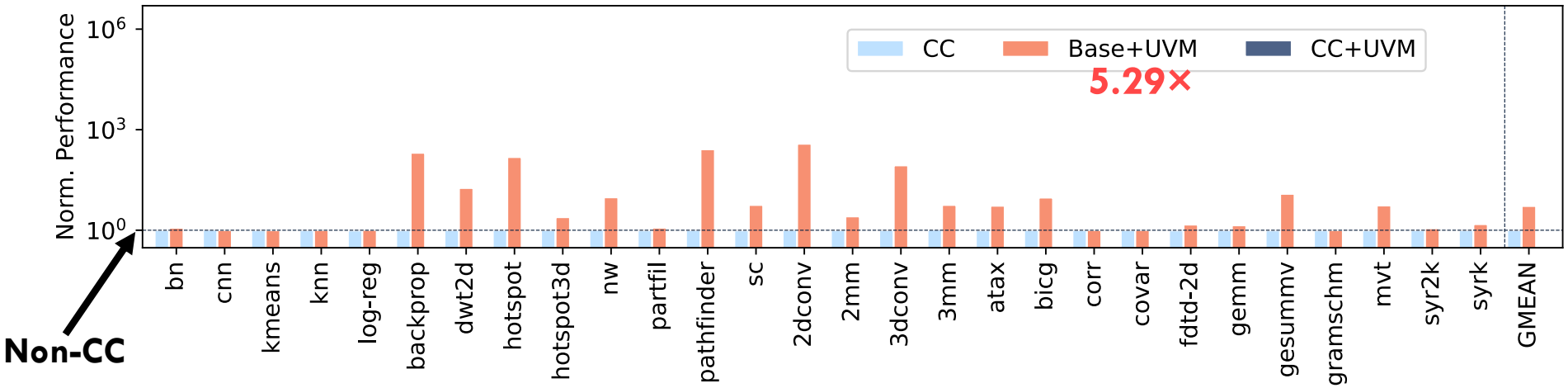
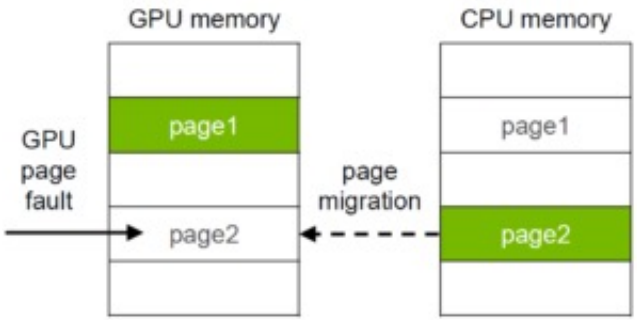


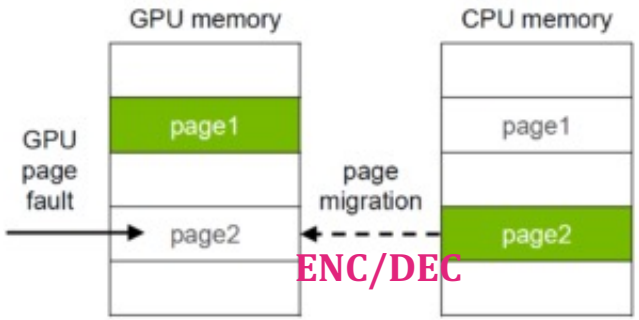
Fig.3 Normalized kernel execution time.

Single Address Space



Source: NVIDIA

Frequent CPU-GPU interaction!



With CC ON

Kernel Execution Time

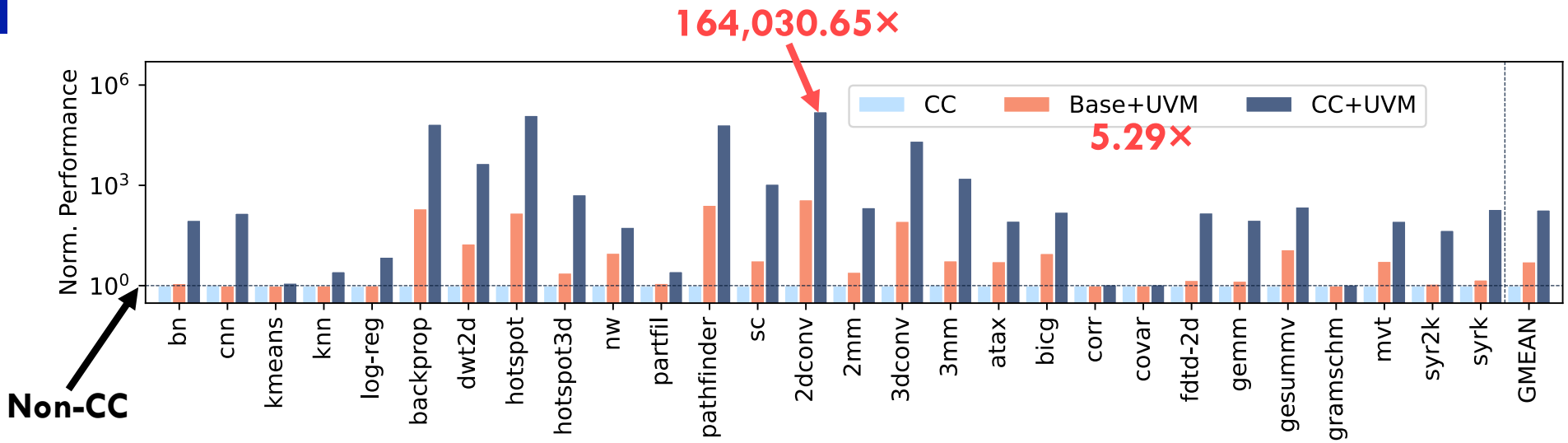


Fig.3 Normalized kernel execution time.

Kernel Execution Time

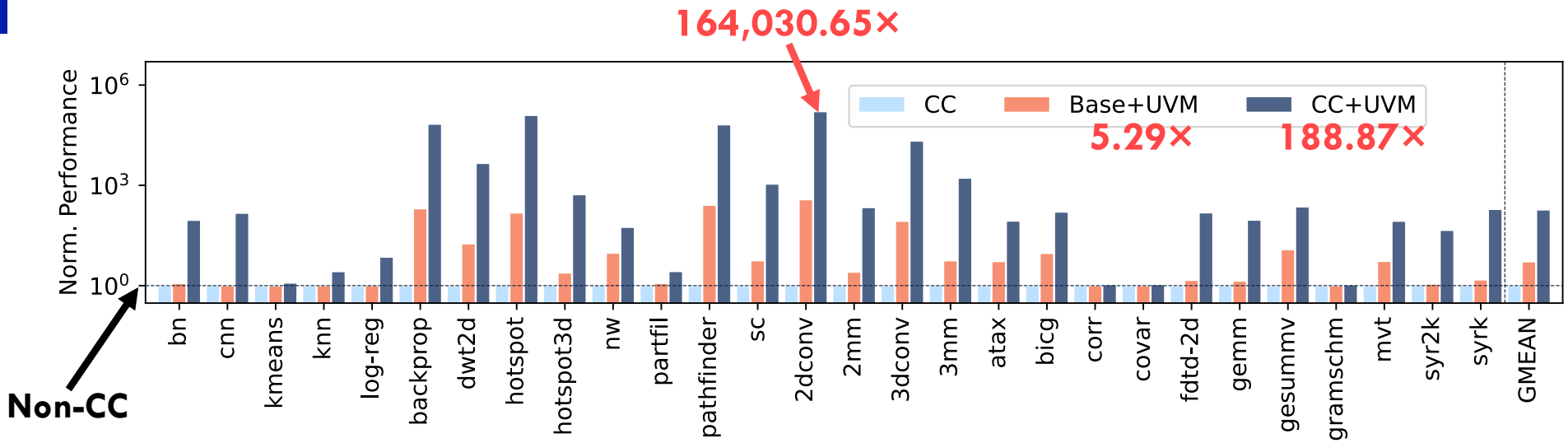


Fig.3 Normalized kernel execution time.

Kernel Execution Time

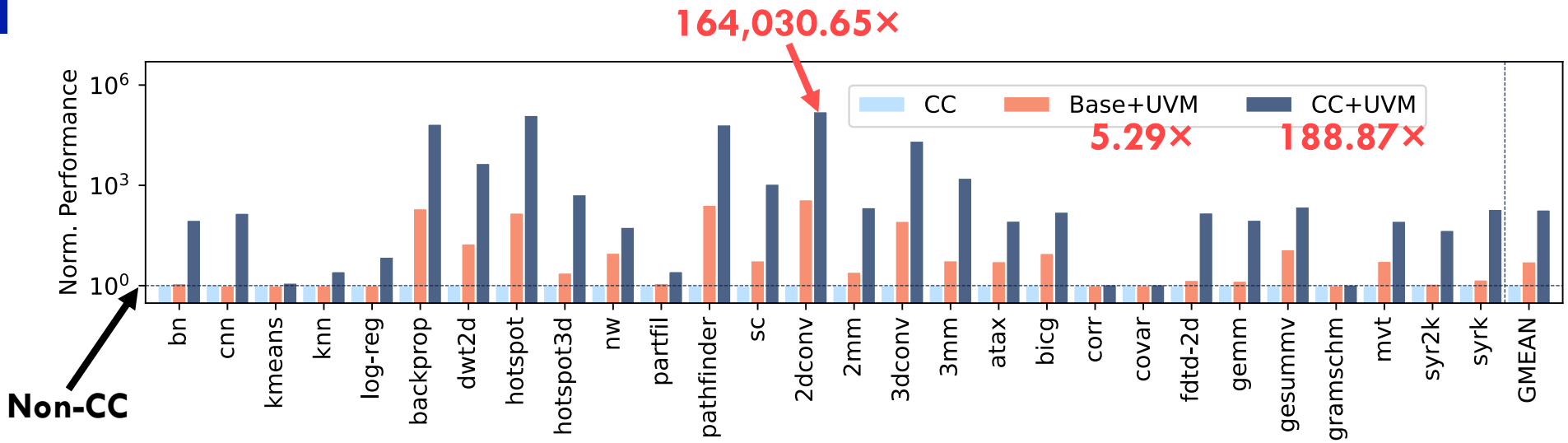


Fig.3 Normalized kernel execution time.

Observation 3.

- CC has **minimal** impact on **non-UVM** kernels (**0.48% increase**).
- **UVM** in CC mode incurs an average slowdown of **188.87×**.

Memory Transfer Time

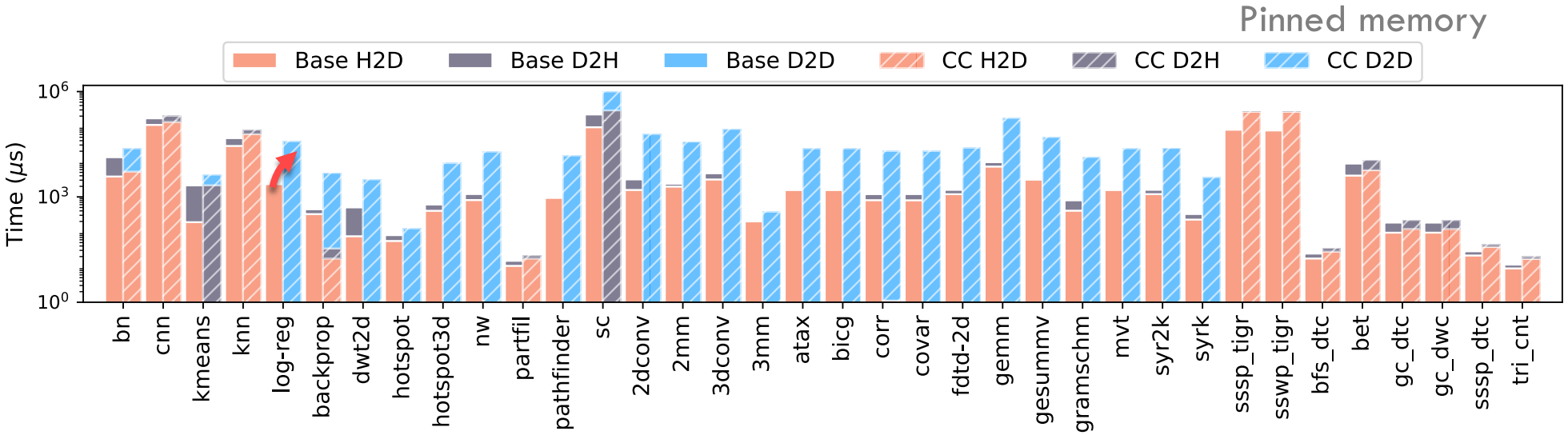


Fig.4 Time (μs) spent on memory copy.

Observation 4.

- ✓ On average, copy operations in CC mode take **5.80×** longer compared to non-CC mode, with a maximum slowdown of **19.69×**.
- ✓ Pinned memory is converted to UVM encrypted paging in CC mode which incurs high overhead.

Launch and Queuing

Results are normalized to non-CC time.

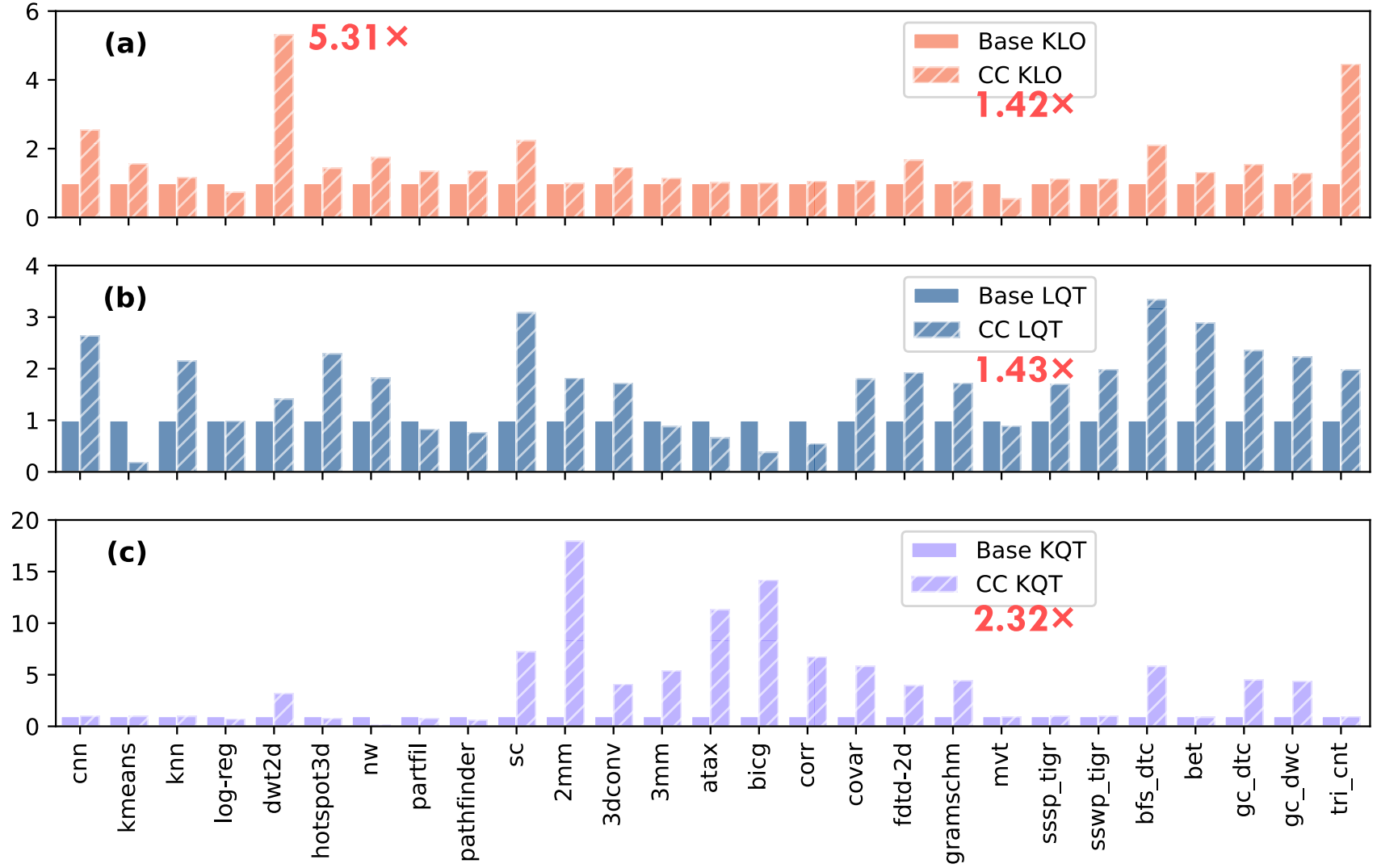


Fig.5 Effect of CC on KLO, LQT, and KQT.

Closer Look

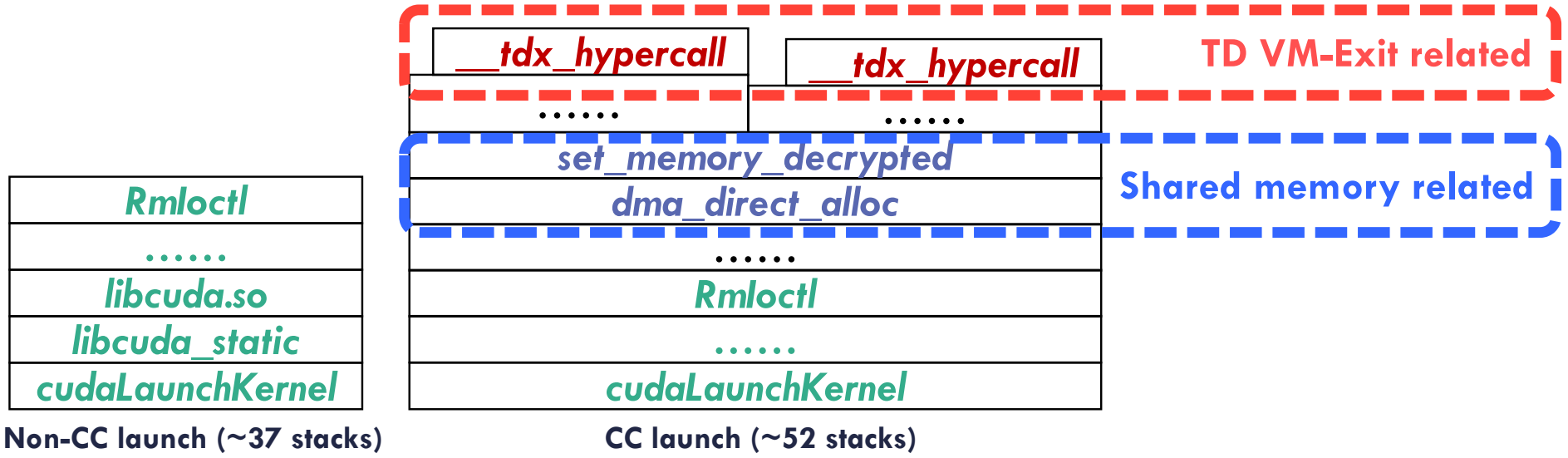


Fig.6 Simplified call stack of a cudaLaunchKernel call, obtained via perf.

Observation 5.

- ✓ On average, CC increases KLO by **1.42×** mostly due to TDX ***hypercalls***. It increases LQT by **1.43×** and KQT by **2.32×**.
- ✓ For applications with a low number of kernel launches, KQT can be significantly amplified.

Case Study

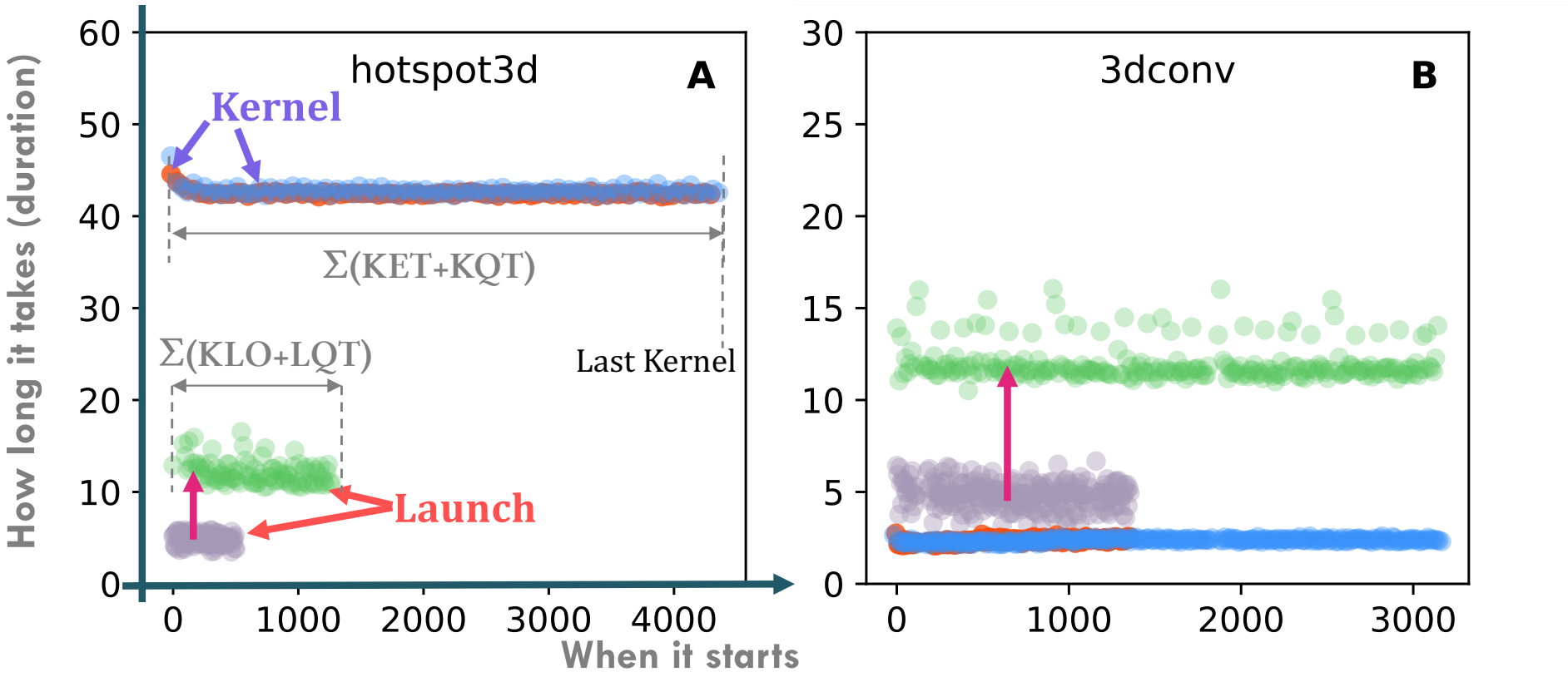


Fig.7 Distribution of events and their durations for representative applications, in μs.

Case Study

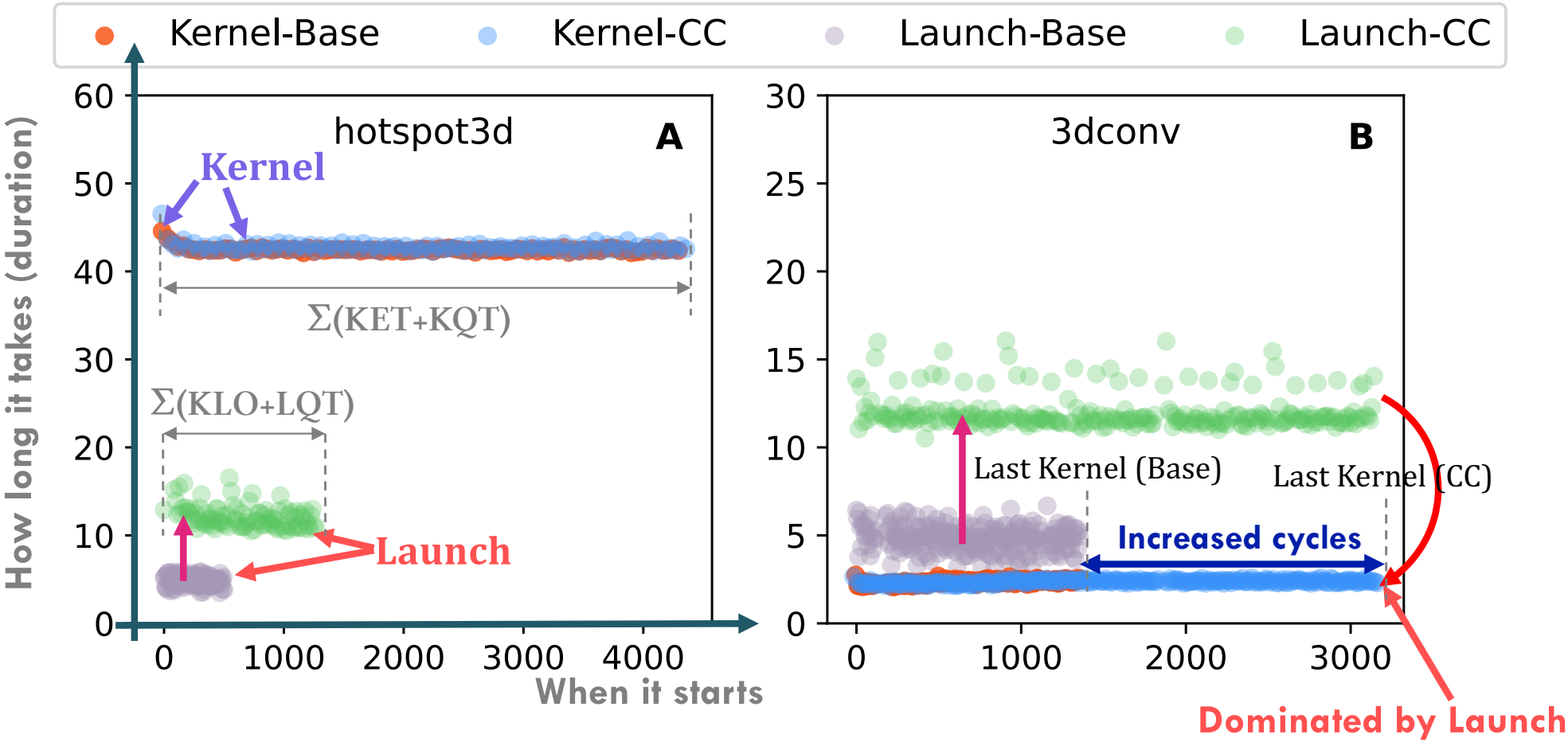


Fig.7 Distribution of events and their durations for representative applications, in μs.

Case Study

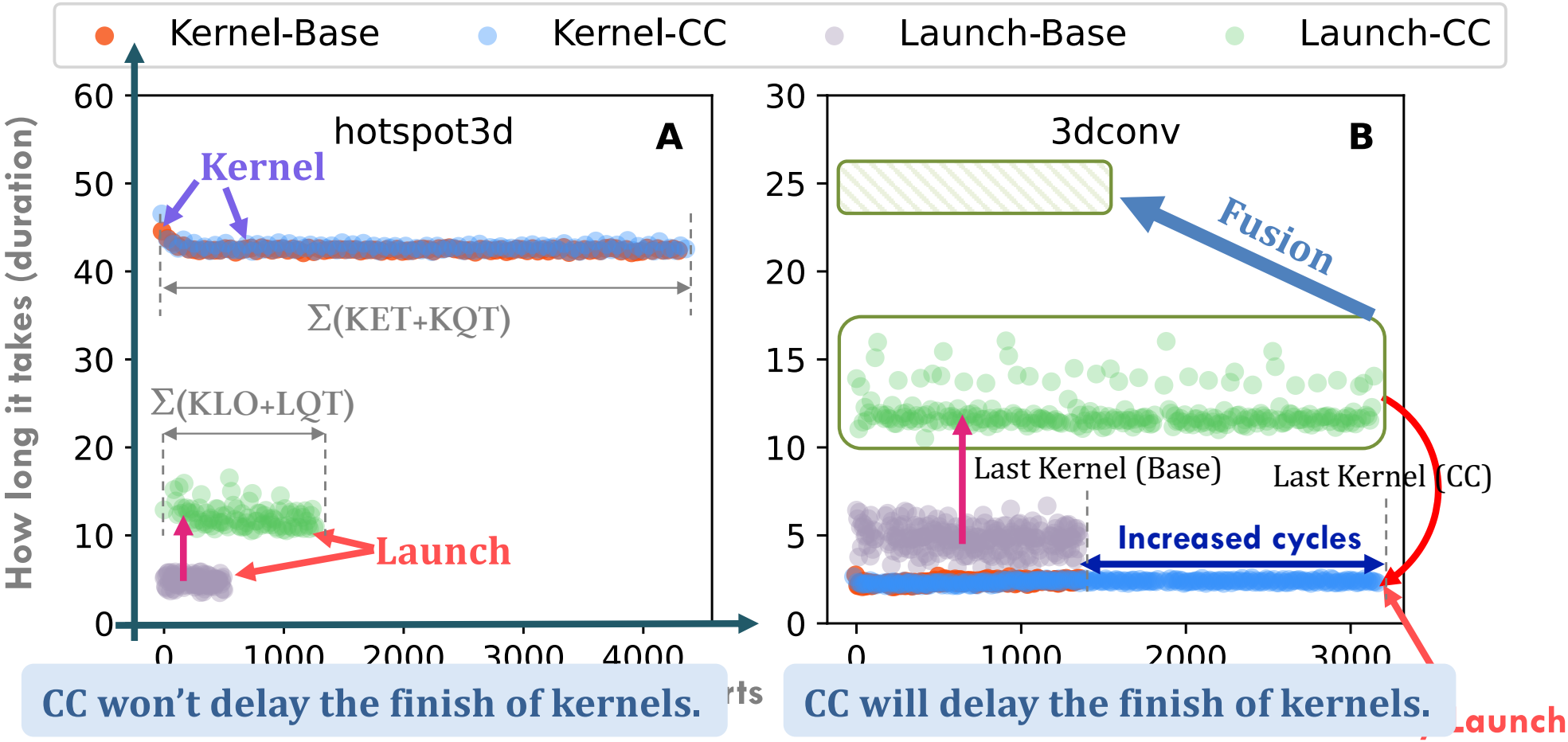
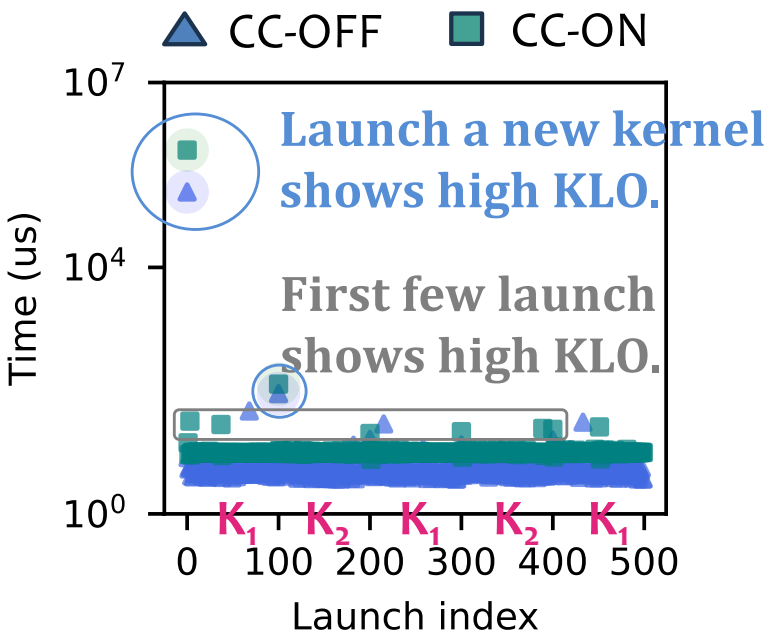
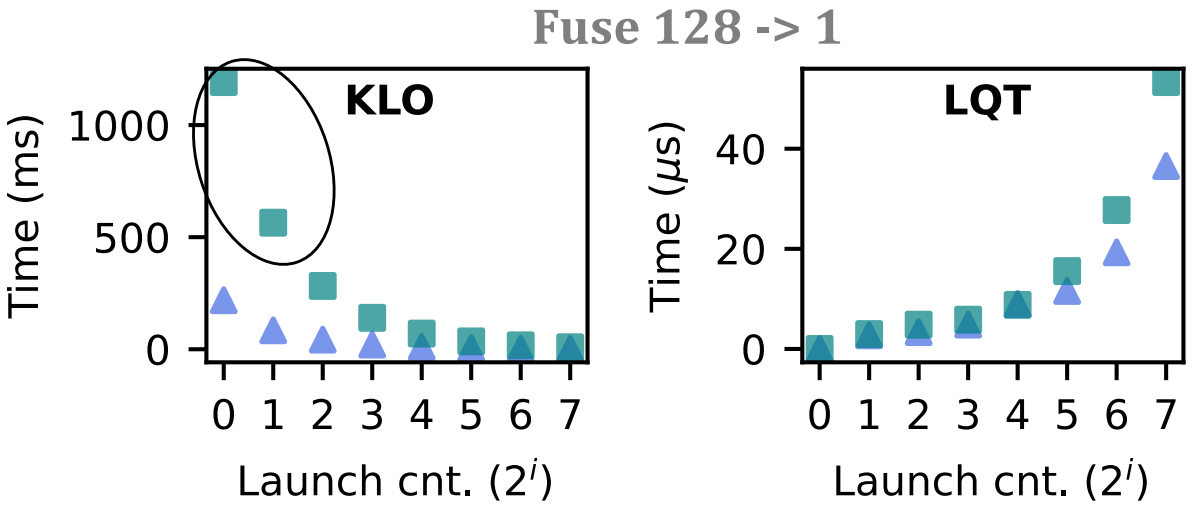


Fig.7 Distribution of events and their durations for representative applications, in μs.

How to optimize? **Insights: Fusion and overlapping!**

Insight: Fusion





Aggressive fusion is not working, a right fusion parameter helps.

More Details

- Memory Management
- Kernel-to-Launch Ratio
- Overlapping
- CNNs
- LLMs
- Quantization
- …

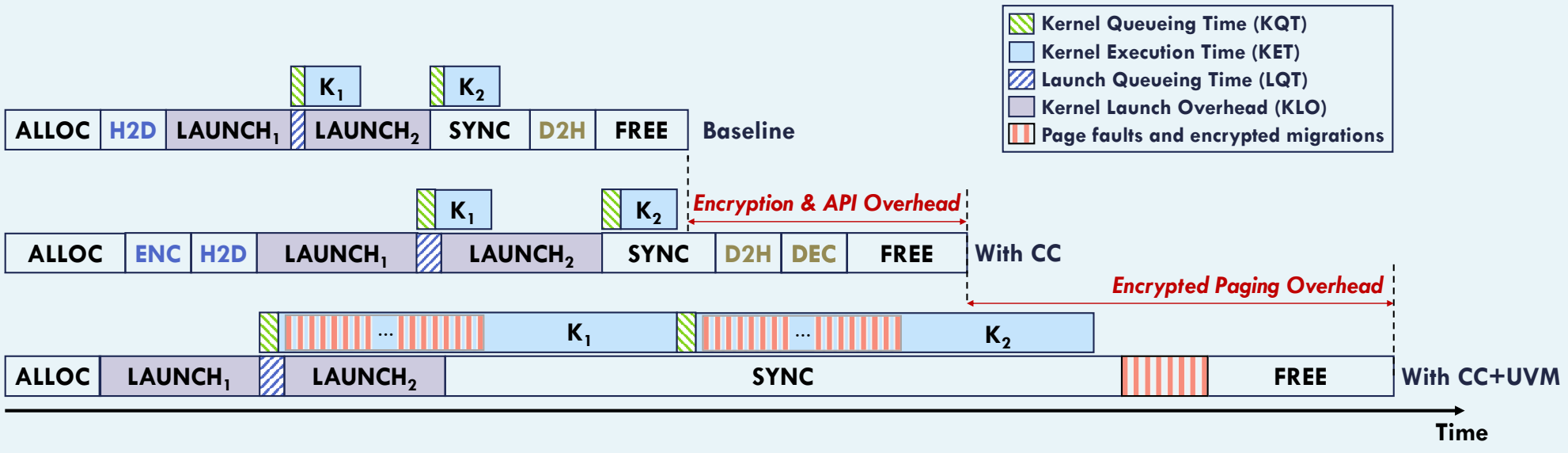
More details in the paper!

Conclusion

We present a **comprehensive performance evaluation** of GPU-based CC guided by a simple performance model, and considered several **optimizations** towards addressing the overheads of CC.

Our results on real hardware show that GPU-based CC comes with:

- Data movement: **5.80× (avg)** **19.69× (max)** **3.03 GB/s**
- Encryption: **3.36 GB/s (AES-GCM)**
- Kernel Execution: **0.48% (CC)** **188.87× (avg, CC+UVM)**
- Kernel Launch: **1.42× (avg)** **5.31× (max)**
- Queuing: **1.43× (avg, LQT)** **2.23× (avg, KQT)**



Thank You!

Questions?

Dissecting Performance Overheads of Confidential Computing in GPU-based Systems

Yang Yang, Mohammad Sonji, Adwait Jog

{yangyang, npv2tk, ajog} @virginia.edu

UNIVERSITY of VIRGINIA