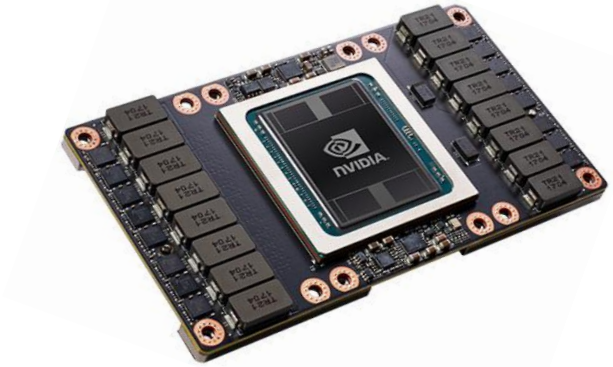
Confidential Accelerators

Stavros Volos
Microsoft Research

Accelerators Play Pivotal Role in Cloud

GPU, FPGA, AI Accelerators (e.g., Brainwave, TPU)



Increasing compute performance and bandwidth

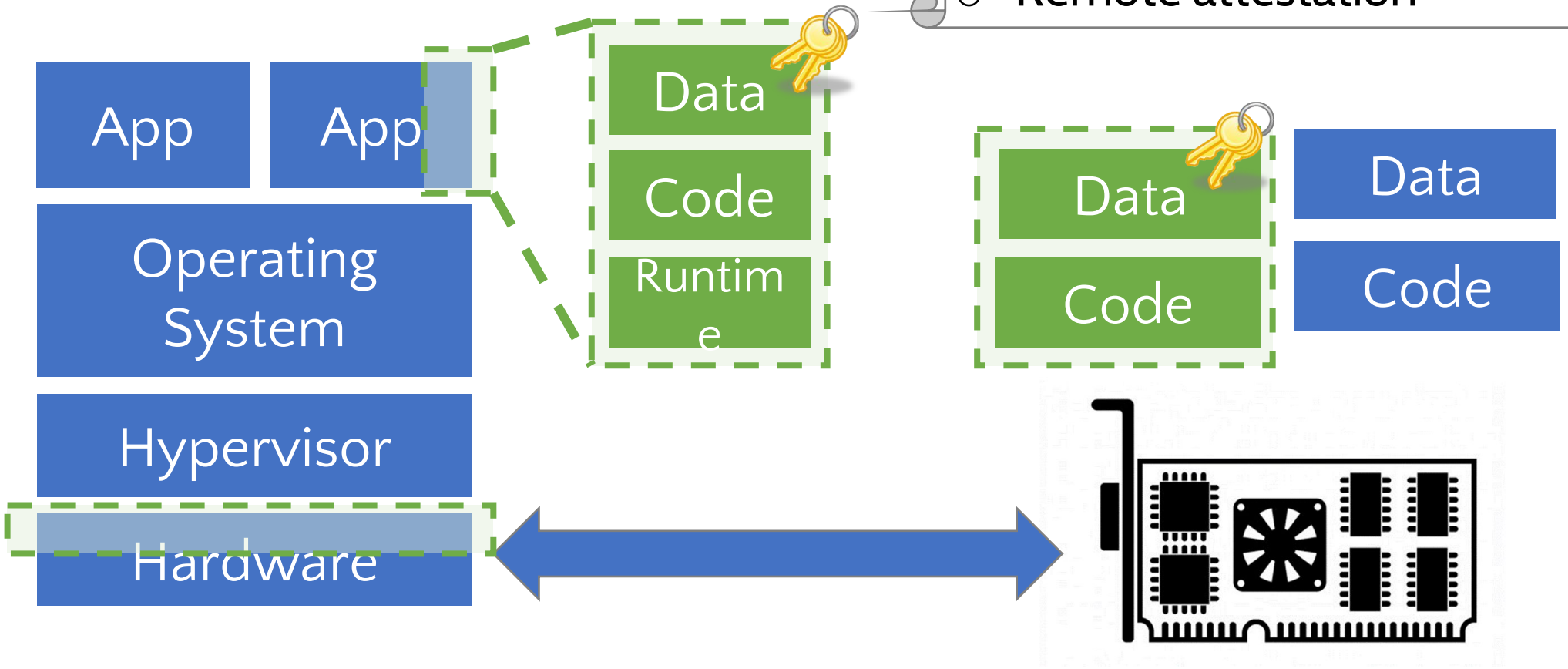
- E.g., Nvidia V100 offers 14 TFLOPS & ~1 TB/s of Mem. BW
- 100x faster than CPU

But, not designed with confidential computing primitives

Undesirable trade-off between security and performance

Confidential Accelerators

- Securely offload code and data
- Isolation from privileged attackers
- Remote attestation

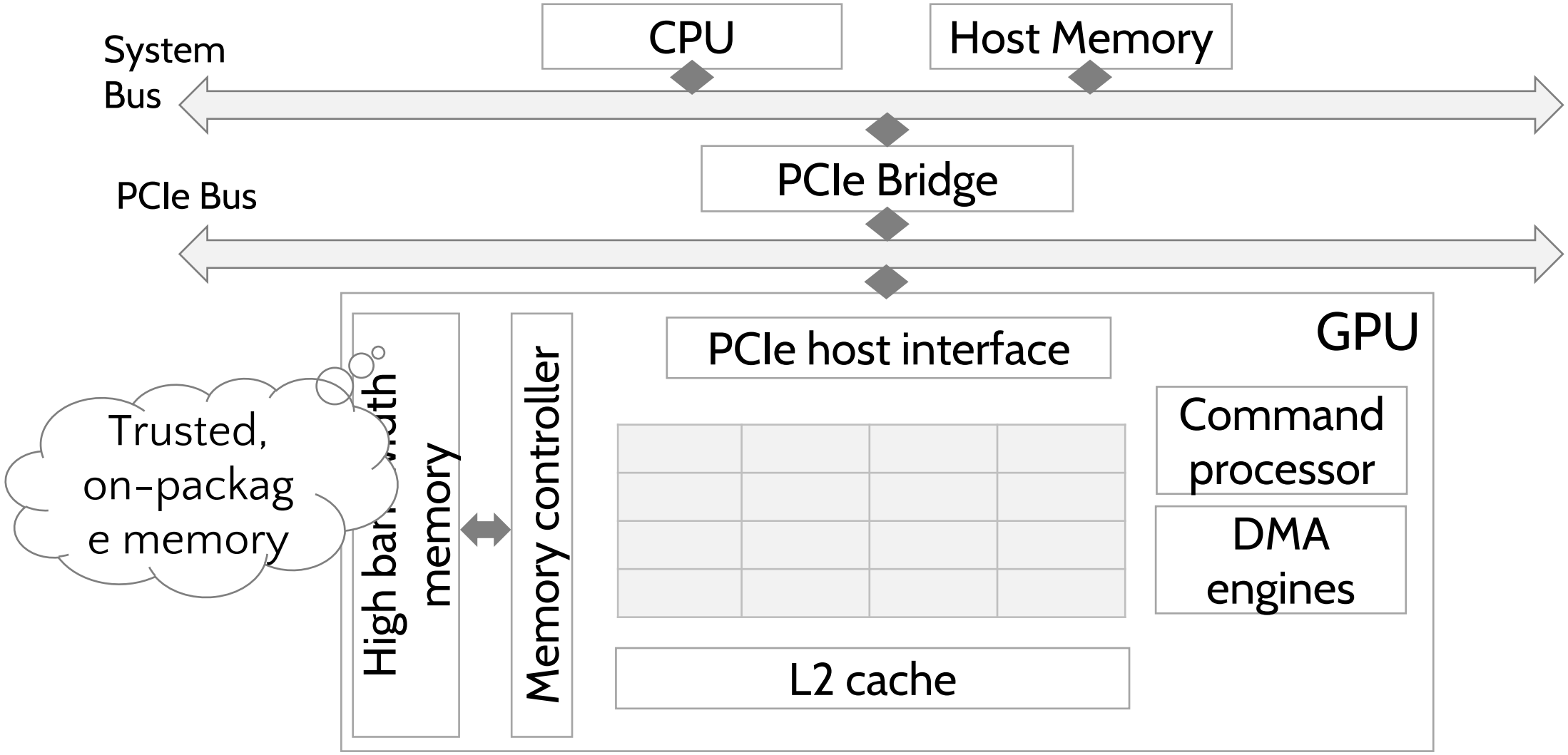


How do we design trusted execution environments (TEE) for accelerators?

Confidential GPUs‡

‡ Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. Graviton: Trusted Execution Environment on GPUs, OSDI'18

GPU 101 – Overview



GPU 101 – Execution Environment

Execution is context-based

- Context is a collection of resources akin to a CPU process
- Resources include command queues and device memory

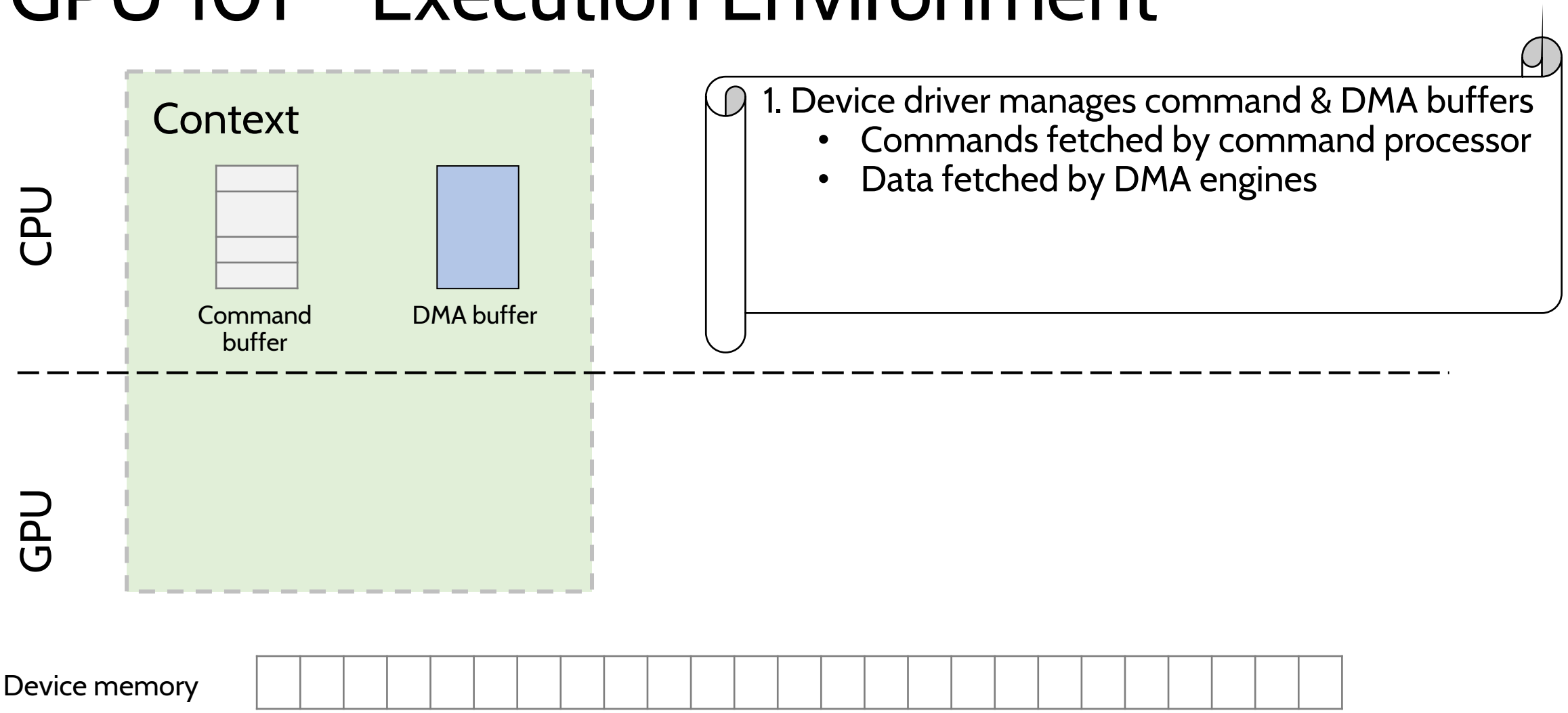
Kernels operate on data in device memory

- Kernel is application code executed on GPU cores via API call
- Device memory management (allocation, transfer) is explicit via API calls

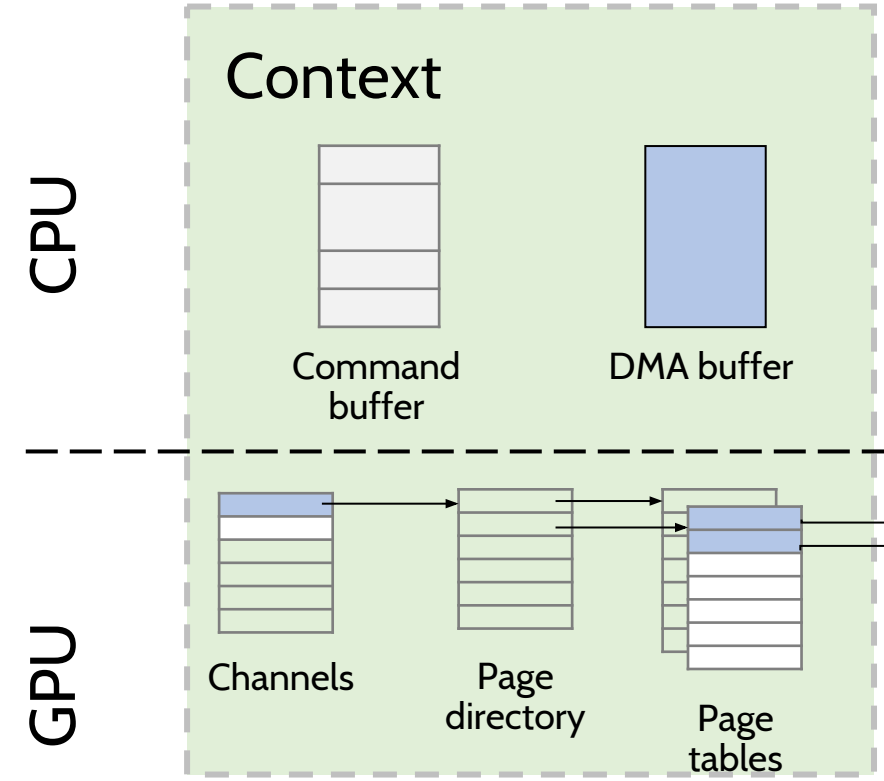
GPU cores and DMA engines controlled via command channels

- Kernel's code/data transfer to device memory
- Kernel dispatch to GPU

GPU 101 – Execution Environment



GPU 101 – Execution Environment



1. Device driver manages command & DMA buffers
   - Commands fetched by command processor
   - Data fetched by DMA engines
2. Device driver controls memory management
   - GPUs rely on virtual memory and channel abstraction for context isolation

Large attack vector if host OS and device driver are

TEE on GPUs (1/2)

Key idea: TEE takes the form of secure context

- Cryptographically bound to user's trusted GPU runtime
- Isolated from all software on host and other contexts
- Commands and DMA buffers receive ciphertext

Modest hardware complexity: extensions limited to command processor

- New commands for context and memory management
  - Provision context's key during context creation
  - Enforce strict context-level ownership of device memory
  - Keyed MACs passed from/to GPU runtime for authorization and validation
- Crypto engine for authenticated encryption

Hardware changes NOT on the critical path of the GPU

TEE on GPUs (2/2)

Software extensions limited to GPU runtime and GPU driver

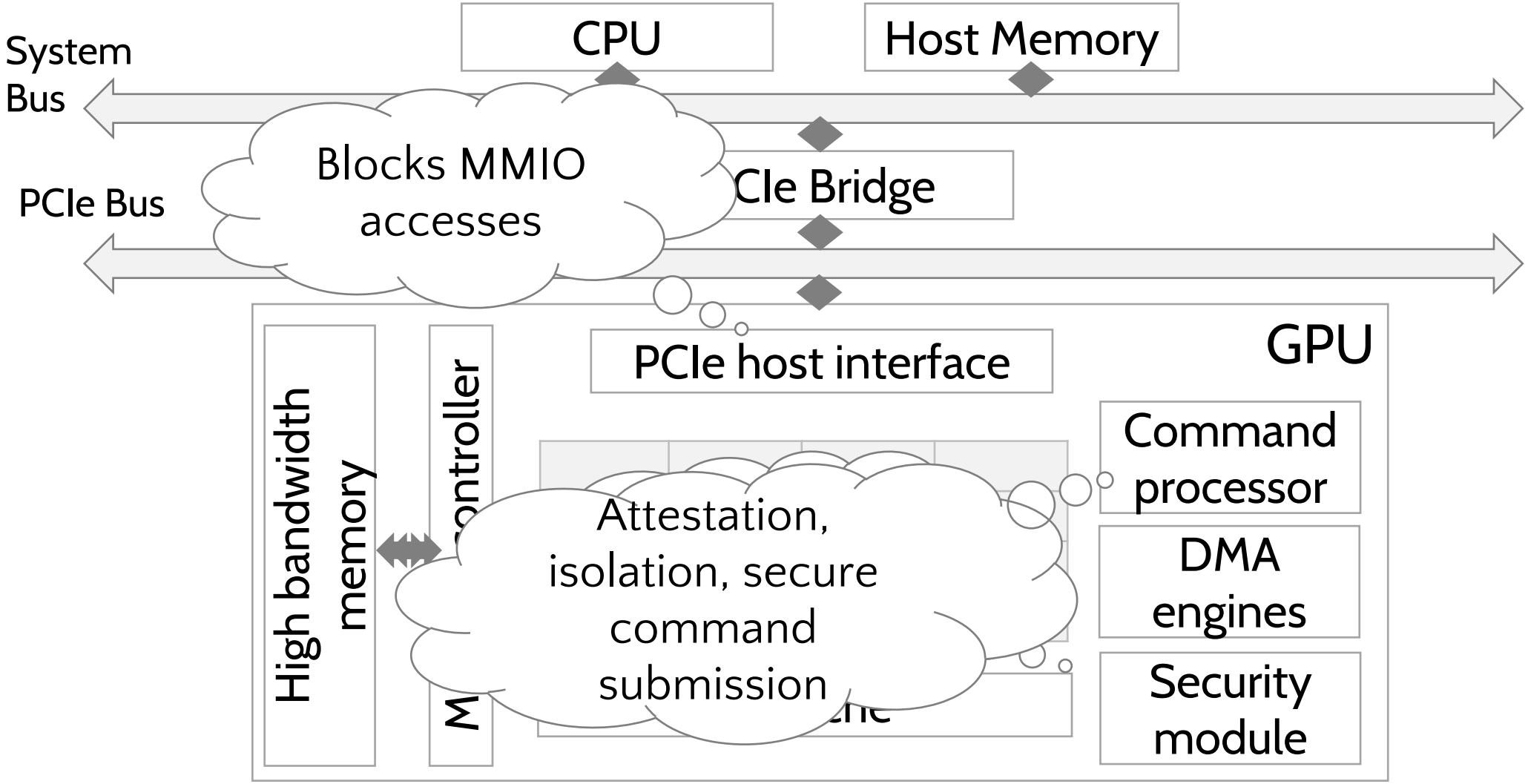
- Secure CUDA driver API implements all required security protocols
- GPU driver uses bootstrap channel for context/memory management
- GPU runtime validates device driver's actions as part of CUDA driver API

Complexity hidden behind CUDA programming model

- CUDA programming model well defined for abstracting device management
- No/minimal modifications to user application codebase

Almost all changes are transparent to application developers

TEE on GPUs – Summary



Summary

Current cloud trends

- Accelerators are the future of the cloud
- No confidential computing support by accelerators

Trusted execution environment for accelerators

- Allow CPU TEEs for secure code/data offload to device
- Can be simpler than CPU TEEs