Graviton
Trusted Execution Environments on GPUs

Stavros Volos,† Kapil Vaswani,† and Rodrigo Bruno‡
†Microsoft Research ‡University of Lisbon
Trends in Cloud Computing

Accelerators play pivotal role in cloud
- CPUs running out of steam due to *End of Moore's Law*
- GPUs, FPGAs, custom silicon deliver 10-100x higher performance

Cloud privacy important but challenging
- Customers operate on sensitive data (e.g., patients, transactions)
- Increasing frequency and sophistication of data breaches

Need strong security mechanisms for preserving data privacy in cloud
Confidential Cloud Computing

Trust-execution Environments (TEE)
• Execution isolated from privileged attackers
• Remote attestation for establishing trust
• Examples: Intel SGX, ARM TrustZone
• Supported by major cloud providers (e.g. Azure Confidential Computing)

But, CPU TEEs cannot be used in apps that utilize accelerators

Undesirable trade-off between performance and security
Our Proposal: Graviton

Graviton: Trusted Execution Environments on GPUs

- Execution isolated from system software and other co-tenants
- Remote attestation for establishing trust

Contributions

- Graviton architecture with minimal hardware extensions
- Extensions to CUDA runtime for end-to-end security
- Graviton implementation for demonstrating low performance overheads
Outline

• Introduction
• GPUs & Threat Model
• Graviton
• Evaluation
• Conclusion
GPU 101: System Stack

GPU engines controlled via group of commands
• Generated by runtime and fetched by command processor
GPU 101: Execution Model

Contexts supported by channels
- Implement *virtual memory* abstraction
- Expose command queues to runtime
GPU 101: Cat Classifier Example

1. `cudaMalloc`
2. `cudaMemcpy`
3. `cudaLaunch`

---

**Application > Runtime:**

1. Request
2. Allocation
3. Launch

---

**OS / Driver:**

- Code
- Data

---

**Hypervisor:**

- MMIO

---

**CPU > GPU:**

- PCI Bus

---

**Host:**

- Context
- Command buffer
- DMA buffer

---

**Device:**

- Channel descriptor
- Page directory
- Page tables

---

**Memory:**

- Allocation

---

**Application:**

- GPU
- Cat Classifier Example

---

**GPU Architecture:**

- Host to Device
- Device to Host
GPU 101: Tampering with Commands & Data

Context 1

Host

Device

Memory

Command buffer DMA buffer

Channel descriptor Page directory Page tables

Malicious OS
GPU 101: Violating Context Isolation

Context 1
- Command buffer
- Channel descriptor
- Page directory
- Page tables

Context 2
- Command buffer
- Channel descriptor
- Page directory
- Page tables

Malicious OS

Host

Device

Memory
Threat Model

Trusted computing base
- GPU package including on-package memory
- CPU package including TEE implementation
- GPU runtime hosted in CPU TEE

Goal: Confidentiality and integrity of computation and data

Out of scope: side channels and package assembly attacks
Outline

• Introduction
• GPUs & Threat Model
• Graviton
• Evaluation
• Conclusion
Graviton: Overview

Key concept: Redefined interface between hardware and software

Hardware primitives in GPU
  • Remote attestation for establishing trust
  • Context isolation
  • Secure command submission

Runtime abstractions
  • Secure memory management
  • Secure memory copy and task launch
Graviton: Context Isolation

Protected memory
• Hosts VM structures, code, and data
• CPU’s MMIO accesses are blocked

Virtual memory management via CP
• Ensures use of protected memory
• Exclusive use of context’s memory resources

Secure command submission
• Session key during context creation
• Only owner runtime can execute tasks
Graviton: Secure Memory Copy

Key concept
• Data/code plaintext only inside TEEs
• Data/code ciphertext outside TEE (DMA buffer)

Protocol
Graviton: Secure Memory Copy

Key concept
- Data/code plaintext only inside TEEs
- Data/code ciphertext outside TEE (DMA buffer)

Protocol
- Secure submission of *copy* task
Graviton: Secure Memory Copy

Key concept
• Data/code plaintext only inside TEEs
• Data/code ciphertext outside TEE (DMA buffer)

Protocol
• Secure submission of *copy* task
• Secure submission for *authenticated decryption*
Graviton in a Nutshell

Low hardware complexity
• Changes limited to peripheral components
• No changes to CPU, GPU cores and memory

Transparent to developers
• GPU runtime abstractions
• Hidden behind GPU programming model

Remote attestation
VM mgmt. commands
AES-GCM engine

Endorsement key
ECDSA

Blocks MMIO to Protected Memory

On-package Memory

<table>
<thead>
<tr>
<th>Compute Eng.</th>
<th>PCI Ctrl.</th>
<th>Sec. Mod.</th>
<th>Command Processor</th>
<th>DMA Engines</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
</tr>
<tr>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
</tr>
<tr>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
</tr>
<tr>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
<td>CORE</td>
</tr>
</tbody>
</table>
Implementation

NVIDIA GTX Titan Black
• 2880 CUDA cores, 6GB of memory, peak performance 5.6 TFLOPS

Prototype
• GPU runtime: secure task submission and secure memory management
• Device driver: address-space mgmt. command submission
• Hardware primitives: emulation of new commands and crypto in device driver

Benchmarks: Cifar10-CNN and MNIST-autoencoder
Implications on System Performance

Overhead correlates with ratio between computation and I/O

<table>
<thead>
<tr>
<th></th>
<th>Cifar10</th>
<th>MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insecure</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Isolation</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>Secure Copy</td>
<td>5%</td>
<td>5%</td>
</tr>
</tbody>
</table>

- **Isolation**
  - Secure context management
  - Secure command submission

- **Secure copy**
  - Host-side authenticated encryption
  - GPU-side authenticated encryption
Concluding Remarks

Cloud trends in collision
• Confidentiality and hardware acceleration
• But, confidential computing restricted to CPUs

Graviton: Trusted Execution Environments on GPUs
• Low hardware complexity
• Low performance overheads
• Hardware complexity hidden by GPU programming model