Border Control: Sandboxing Accelerators

Lena E. Olson, Jason Power, Mark D. Hill, David A. Wood

University of Wisconsin-Madison

MICRO-48
December 8th, 2015
Executive Summary

- Accelerators access shared host memory
  + Performance, programmability
  - Bugs, malicious design?

- Protect host from accelerator wild reads/writes

- Border Control
  - provides full host memory safety (sandboxing)
  - does not degrade performance

- Performance overhead ~0.48%
What is an accelerator?

Broadly:

**Specialized hardware** that can perform a **subset of computation tasks** with **higher performance** and/or **lower energy** than a CPU.
Accelerators are Pervasive

- (GP)GPUs
- ISPs
- DSPs
- Cryptographic
- Neuromorphic
- Approximate
- Database
- .....
Accelerators are Programmable

- HSA Model: host, accelerator share memory

- Shared Physical Memory
  - avoid copying data

- Shared Virtual Memory
  - pointer-is-a-pointer semantics
  - improved programmability
Untrusted Accelerators

- May be designed by 3rd parties
- May have bugs
  - Even CPUs have bugs sometimes!
- May be malicious

An incorrect accelerator with access to shared physical memory is a threat!
Threat Model

Protect host from incorrect or malicious accelerators that could perform

- stray reads, violating confidentiality
- stray writes, violating integrity

of host processes that **do** and **do NOT** run on the accelerator
Principle of Least Privilege

Every program and every user of the system should operate using the **least** set of **privileges necessary** to complete the job. Primarily, this principle **limits** the **damage** that can result from an accident or error.

Jerome Saltzer, 1975

Border Control Authors, 2015

... Since GPUs are designed to directly access system memory, and since hardware has historically been considered trusted, it's difficult to ensure all the settings to keep it contained are set accurately, and difficult to ensure whether such settings even work. ...

The most interesting challenge here is protecting against PCIe's Address Translation Services (ATS). Using this feature, any device can claim it's using an address that's already been translated, and thus bypass IOMMU translation. For trusted devices, this is a useful performance improvement. For untrusted devices, this is a big security threat. ATS could allow a compromised device to ignore the IOMMU and write to places it shouldn't have access to.
Outline

Motivation

Border Control

Current Systems

Evaluation

[Diagram of CPU, Accelerator, and Memory or Shared LLC]

[Bar chart showing percentages: 80%, 253%, 82%, 178%, 55%]
Direct Physical Address

- CPU
- TLB
- MMU
- Memory or Shared LLC
- Accel.

- Trusted data path
- Address translation path
- Untrusted data path
- Translation update path
Full IOMMU

- CPU
- TLB
- MMU
- Accel.
- Memory or Shared LLC

- Trusted data path
- Untrusted data path
- Address translation path
- Translation update path
Bypassable IOMMU (Baseline)

CPU

TLB

$$

$$

MMU

Mem req: Virtual addr = V

TLB

$$

$$

Mem req: Phys. addr = P

OS Memory (Q)

Memory or Shared LLC

Process Memory (P)

Trust data path

Address translation path

Untrusted data path

Translation update path
Bypassable IOMMU (Baseline)

- CPU
  - TLB
  - MMU
  - Memory or Shared LLC
  - OS Memory (Q)
  - Process Memory (P)

- Accel.
  - TLB
  - $$

- Mem req: Phys. addr = Q

- Address translation path
- Translation update path

- Trusted data path
- Untrusted data path
Outline

Motivation

Border Control

Current Systems

Evaluation
Bypassable IOMMU (Baseline)

- CPU
- TLB
- MMU
- Accel.
- TLB
- IOMMU

- OS Memory (Q)
- Process Memory (P)
- Memory or Shared LLC

- Trusted data path
- Address translation path
- Untrusted data path
- Translation update path
Border Control

- CPU
- Trusted data path
- Untrusted data path
- TLB
- MMU
- Accel.
- Address translation path
- Translation update path
- Memory or Shared LLC
- OS Memory (Q)
- Process Memory (P)
- IOMMU
Border Control

- CPU
- TLB
- MMU
- $\$\$
- OS Memory (Q)
- Memory or Shared LLC
- Process Memory (P)
- Accele.
- $\$\$
- TLB
- Address translation path
- Translation update path

Trusted data path
Untrusted data path

Mem req: Virtual addr = V
Mem req: Phys. addr = P
Mem req: Phys. addr = P
Border Control

- CPU
- TLB
- Memory or Shared LLC
- OS Memory (Q)
- Process Memory (P)
- Trusted data path
- Untrusted data path
- Address translation path
- Translation update path

Mem req: Phys. addr = Q

IOMMU

Accel.
Border Control: Implementation

- One Border Control instance per accelerator

- Protection Table
  - In system memory
  - Contains all needed permissions by PPN
  - Sufficient for correct design
  - 0.006% physical memory overhead

- Border Control Cache (BCC)
  - Caches recent permissions
  - A 64 byte entry covers 512 4KB pages
Protection Table Design

- Flat physically indexed table in memory

- 2 bits (R/W) per physical page
  - Initialized to 0 (no permission)
  - Lazily updated on IOMMU translation
  - Checked on all accelerator memory requests

<table>
<thead>
<tr>
<th>PPN</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N-3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>N-2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>N-1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
More details in paper!

- Design of Border Control Cache
- Actions: translation, page table updates, etc.
- IBM CAPI
- Multiprocess accelerators
- Large pages
- .....
Methodology

- GPGPU $\rightarrow$ accelerator safety stress-test
- Simulator: gem5-gpu
  - Moderately-threaded: single core
  - Highly-threaded: eight cores
- Rodinia Benchmarks
- Baseline: fast but unsafe bypassable IOMMU
Border Control Overheads

Moderately-Threaded GPU

Takeaway: Average 0.48% performance overhead
Border Control Overheads

Highly-Threaded GPU

Takeaway: Average **0.15%** performance overhead
Conclusion

- Accelerators pose new security questions¹

- Border Control provides full memory access protection / sandboxing
  - with minimal impact on performance
  - and low storage overhead

---

1. “Security Implications of Third-Party Accelerators” by Olson, Sethumadhavan, and Hill, CAL 2015
Questions?
IBM CAPI

- CPU
- TLB
- MMU
- OS Memory (Q)
- Trusted data path
- Untrusted data path
- Address translation path
- Translation update path
- Accel.
- TLB
- Memory or Shared LLC
- IOMMU
- Process Memory (P)
TLB Shootdown Steps

- If page was read-only:
  - update entry in Protection Table and BCC

- If page was read-write:
  1. Invalidate entry in TLB
  2. Flush dirty blocks from page in accelerator cache
  3. Update entry in Protection Table and BCC
**Simulation Parameters**

<table>
<thead>
<tr>
<th>CPU</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Cores</td>
<td>1</td>
</tr>
<tr>
<td>CPU Caches</td>
<td>64KB L1, 2MB L2</td>
</tr>
<tr>
<td>CPU Frequency</td>
<td>3 GHz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GPU</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores (highly threaded)</td>
<td>8</td>
</tr>
<tr>
<td>Cores (moderately threaded)</td>
<td>1</td>
</tr>
<tr>
<td>Caches (highly threaded)</td>
<td>16KB L1, shared 256KB L2</td>
</tr>
<tr>
<td>Caches (moderately threaded)</td>
<td>16KB L1, shared 64KB L2</td>
</tr>
<tr>
<td>L1 TLB</td>
<td>64 entries</td>
</tr>
<tr>
<td>Shared L2 TLB (trusted)</td>
<td>512 entries</td>
</tr>
<tr>
<td>GPU Frequency</td>
<td>700 MHz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory System</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak Memory Bandwidth</td>
<td>180 GB/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Border Control</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BCC Size</td>
<td>8KB</td>
</tr>
<tr>
<td>BCC Access Latency</td>
<td>10 cycles</td>
</tr>
<tr>
<td>Protection Table Size</td>
<td>196KB</td>
</tr>
<tr>
<td>Protection Table Access Latency</td>
<td>100 cycles</td>
</tr>
</tbody>
</table>

Table 3: Simulation configuration details.
## Comparison of Configurations

<table>
<thead>
<tr>
<th></th>
<th>Safe?</th>
<th>L1 $</th>
<th>L1 TLB</th>
<th>L2 $</th>
<th>BCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATS-only IOMMU</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>N/A</td>
</tr>
<tr>
<td>Full IOMMU</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>N/A</td>
</tr>
<tr>
<td>CAPI-like</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>✓</td>
<td>N/A</td>
</tr>
<tr>
<td>Border Control-noBCC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>—</td>
</tr>
</tbody>
</table>
Border Control Cache

Takeaway: A small (1KB) BCC is sufficient for our workloads
**Takeaway:** Permission downgrades affect performance, but not much
Information Flow Tracking

- Goal: track untrusted information, prevent it from modifying sensitive data / control
  - e.g., prevent buffer overflow in software

- Hardware-assisted techniques: prevent threats from bugs in *software* (same address space) – different threat than Border Control

- Hardware (e.g. Tiwari et al., ISCA 2011) – very powerful technique, but high area/runtime overhead and not transparent to software
Mondriaan

- Replacement for traditional page table + TLB
- Allows fine-grained permissions
- Border Control is independent of the policy for deciding permissions
  - But permission granularity might mean alternate Protection Table organizations are better