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

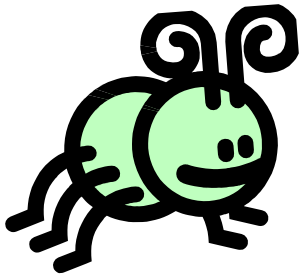
- (GP)GPUs
- ISPs
- DSPs
- Cryptographic
- Neuromorphic
- Approximate
- Database
- .....

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

hardware component

*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

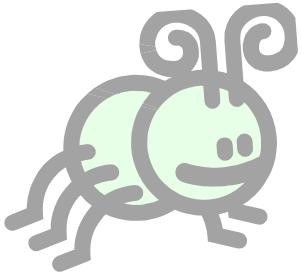
Fuzzing PCI express: … (2/2017)

https://**cloudplatform.googleblog.com**/2017/02/fuzzing-PCI-Express-security-in-plaintext.html?m=1 (added after MICRO'15)

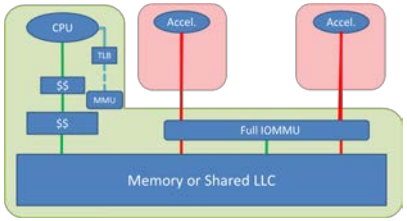
… Since GPUs are designed to directly access system memory, and since hardware has historically been considered trusted, it's difficult to ensure all the settings to keep it contained are set accurately, and difficult to ensure whether such settings even work. …

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

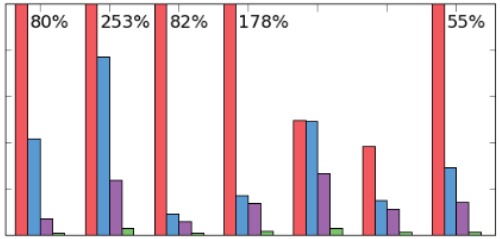


Motivation



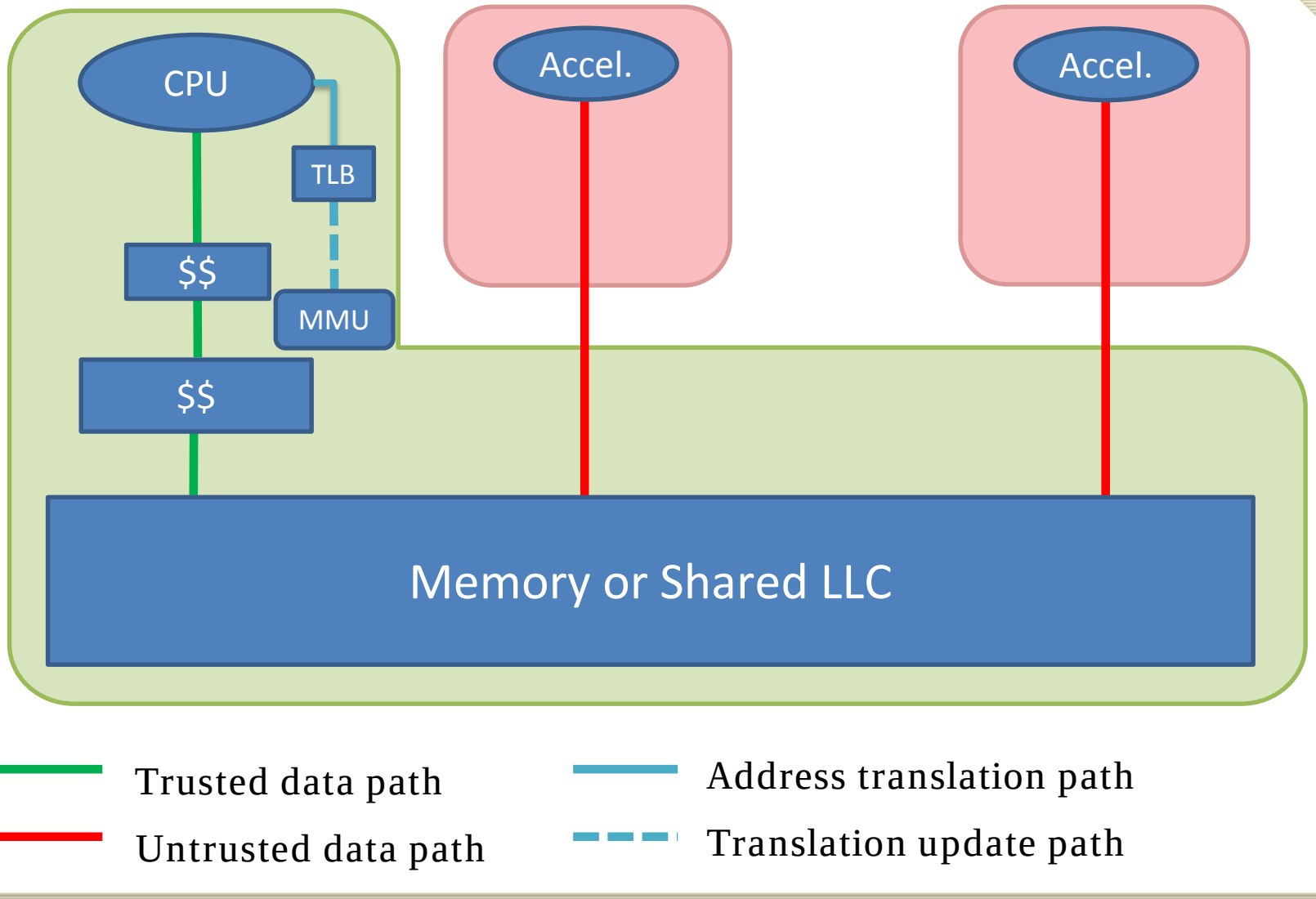
Current Systems

Border Control

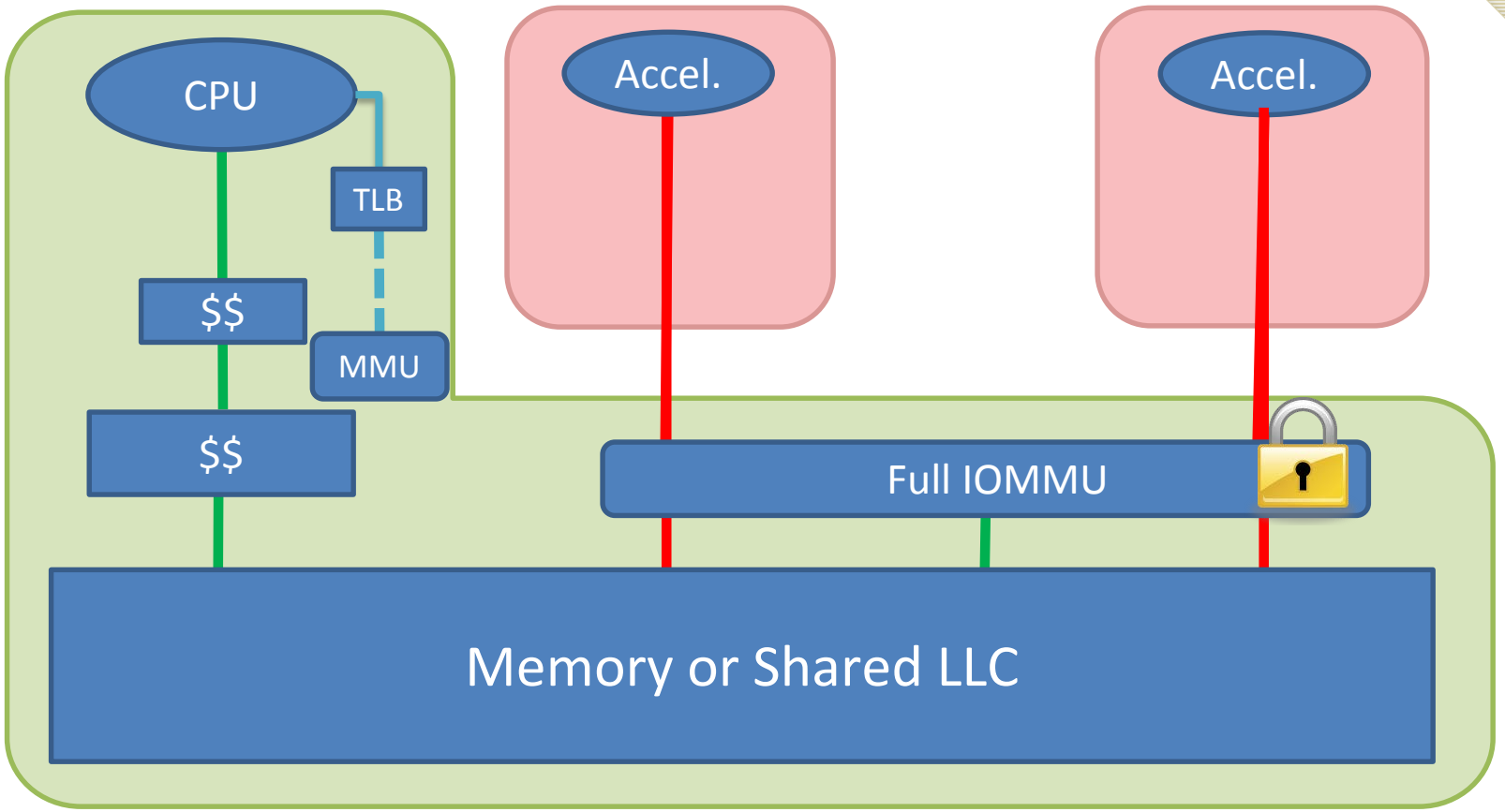


Evaluation

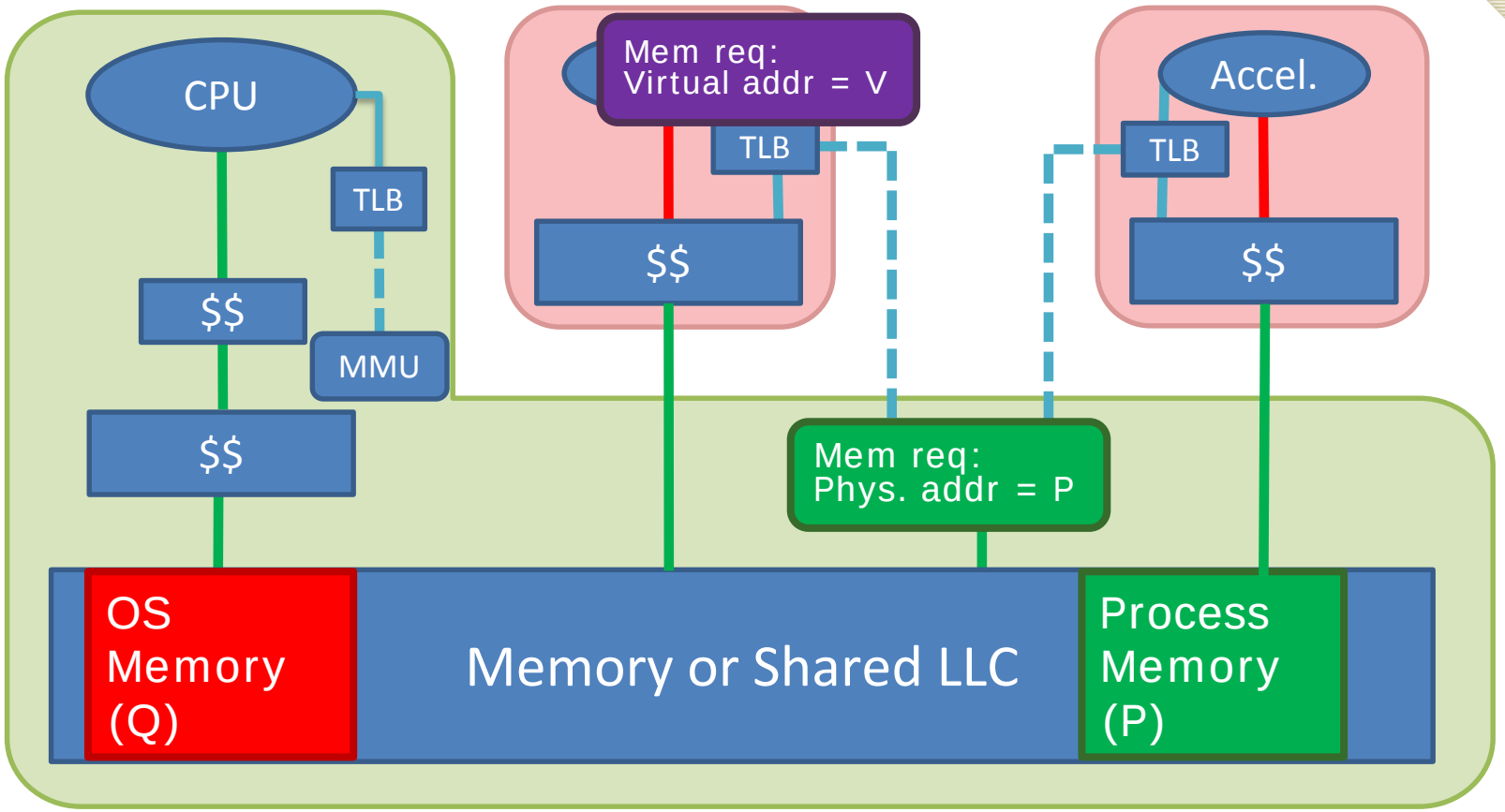
Direct Physical Address



Full IOMMU


CPU
TLB
$$
MMU
$$
Accel.
Accel.
Full IOMMU
Memory or Shared LLC
Trusted data path
Untrusted data path
Address translation path
Translation update path

Bypassable IOMMU (Baseline)



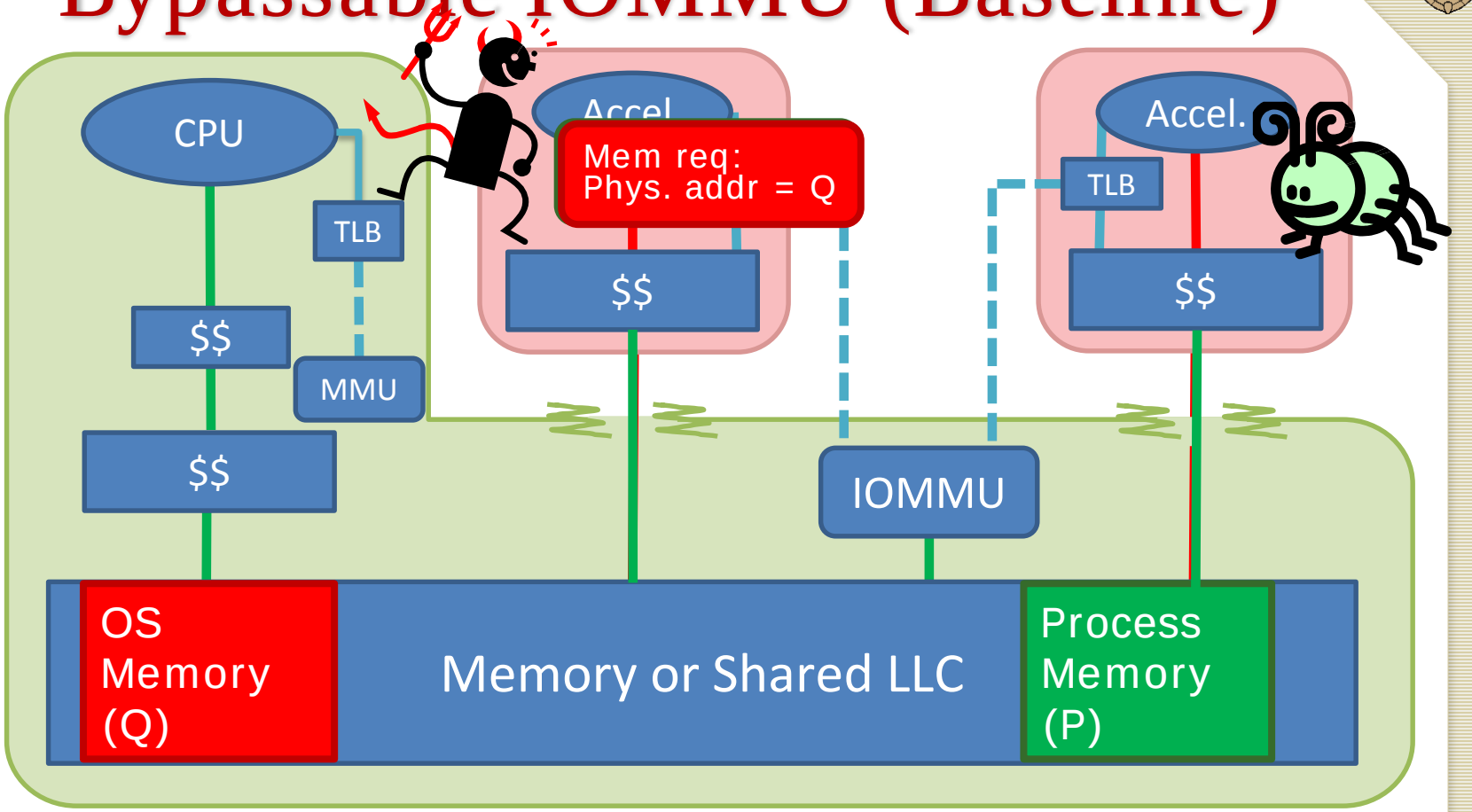
Trusted data path

Untrusted data path

Address translation path

Translation update path

Bypassable IOMMU (Baseline)



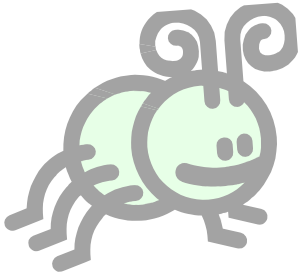
Trusted data path

Untrusted data path

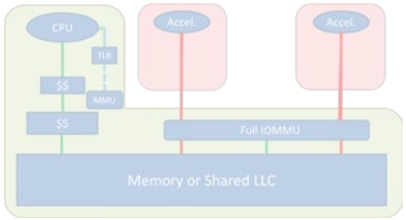
Address translation path

Translation update path

Outline

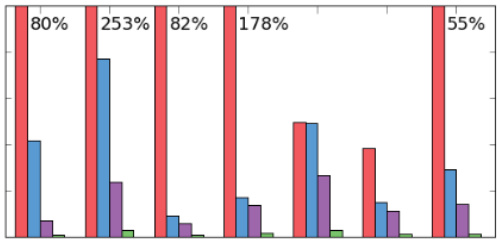


Motivation



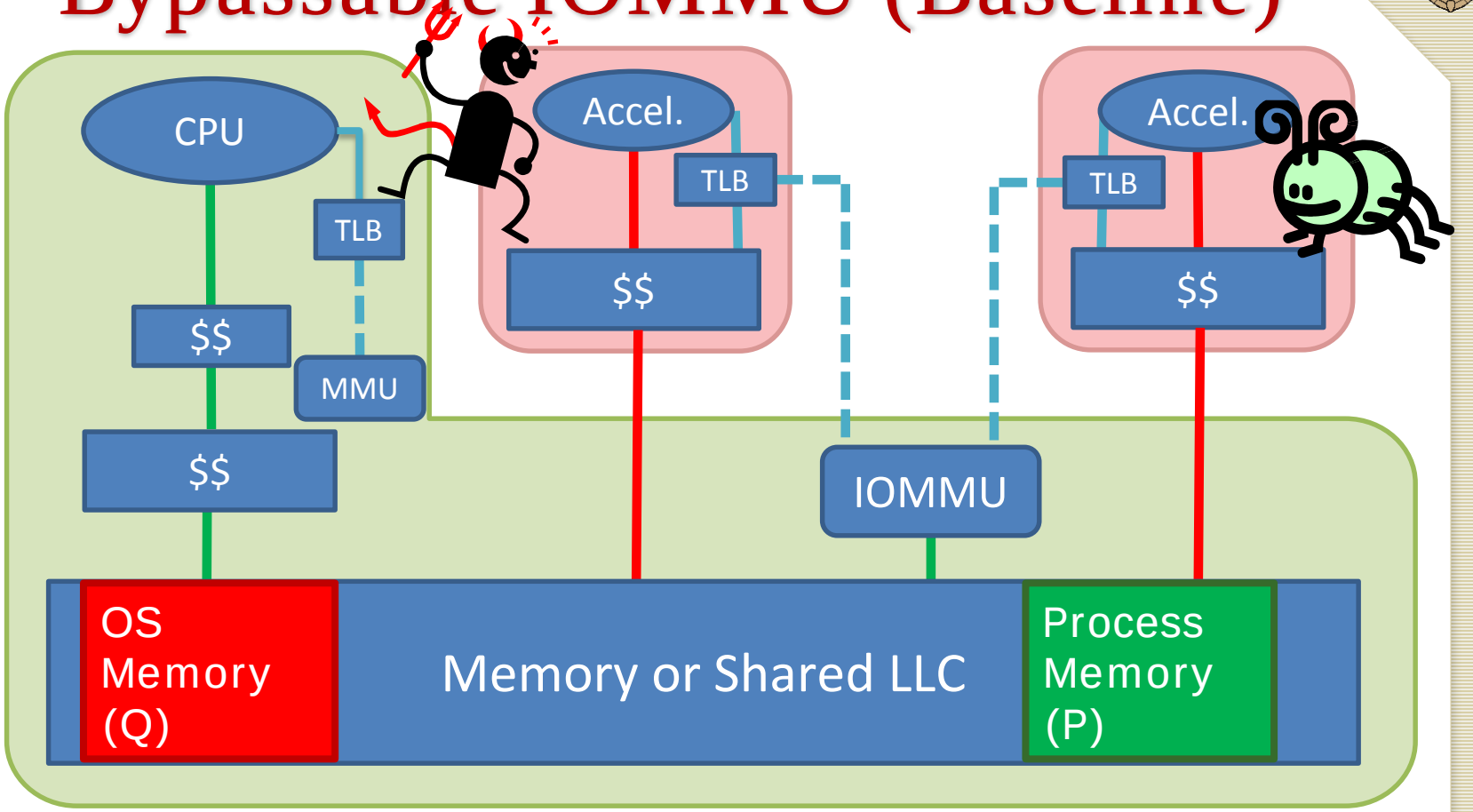
Current Systems

Border Control



Evaluation

Bypassable IOMMU (Baseline)



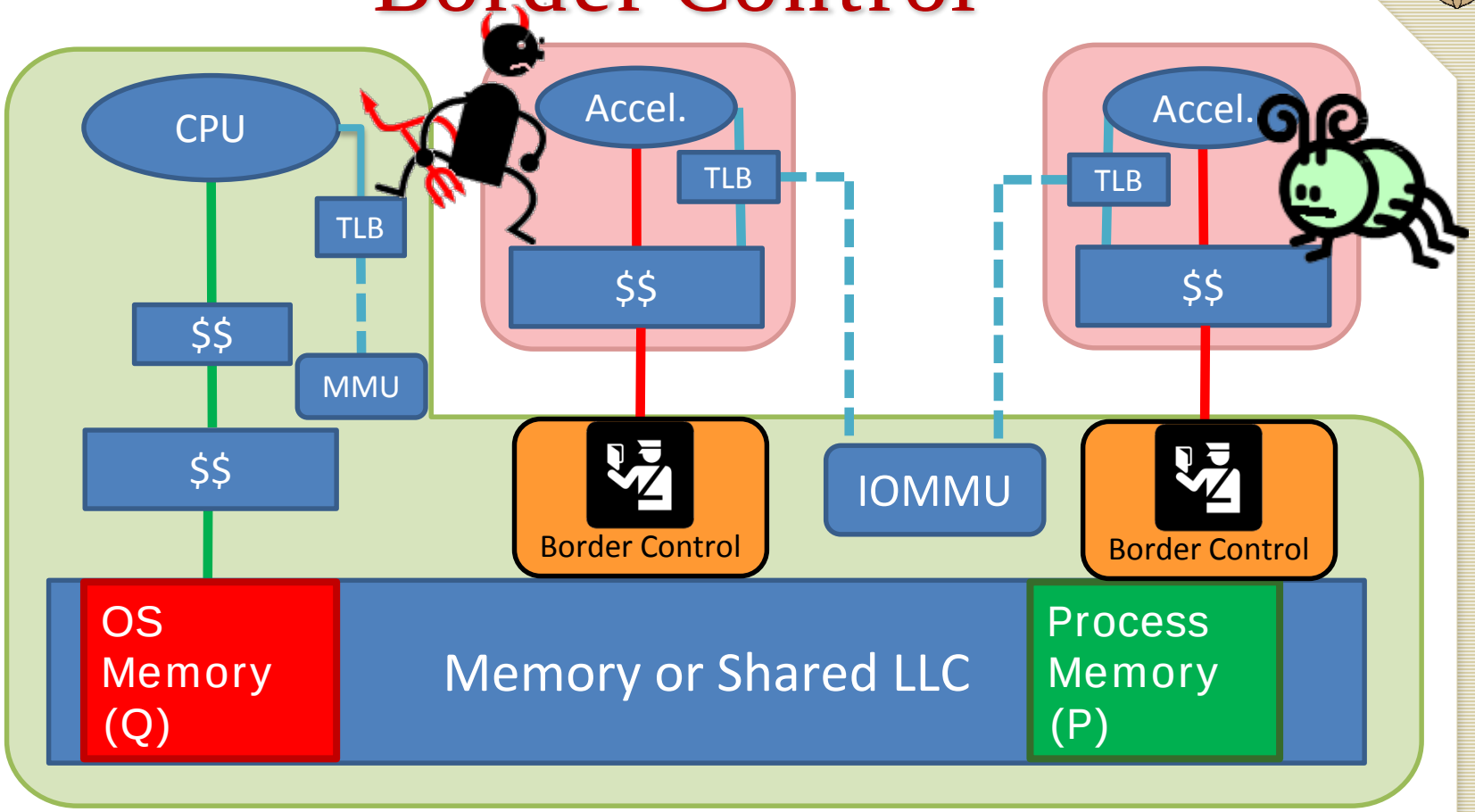
Trusted data path

Untrusted data path

Address translation path

Translation update path

Border Control



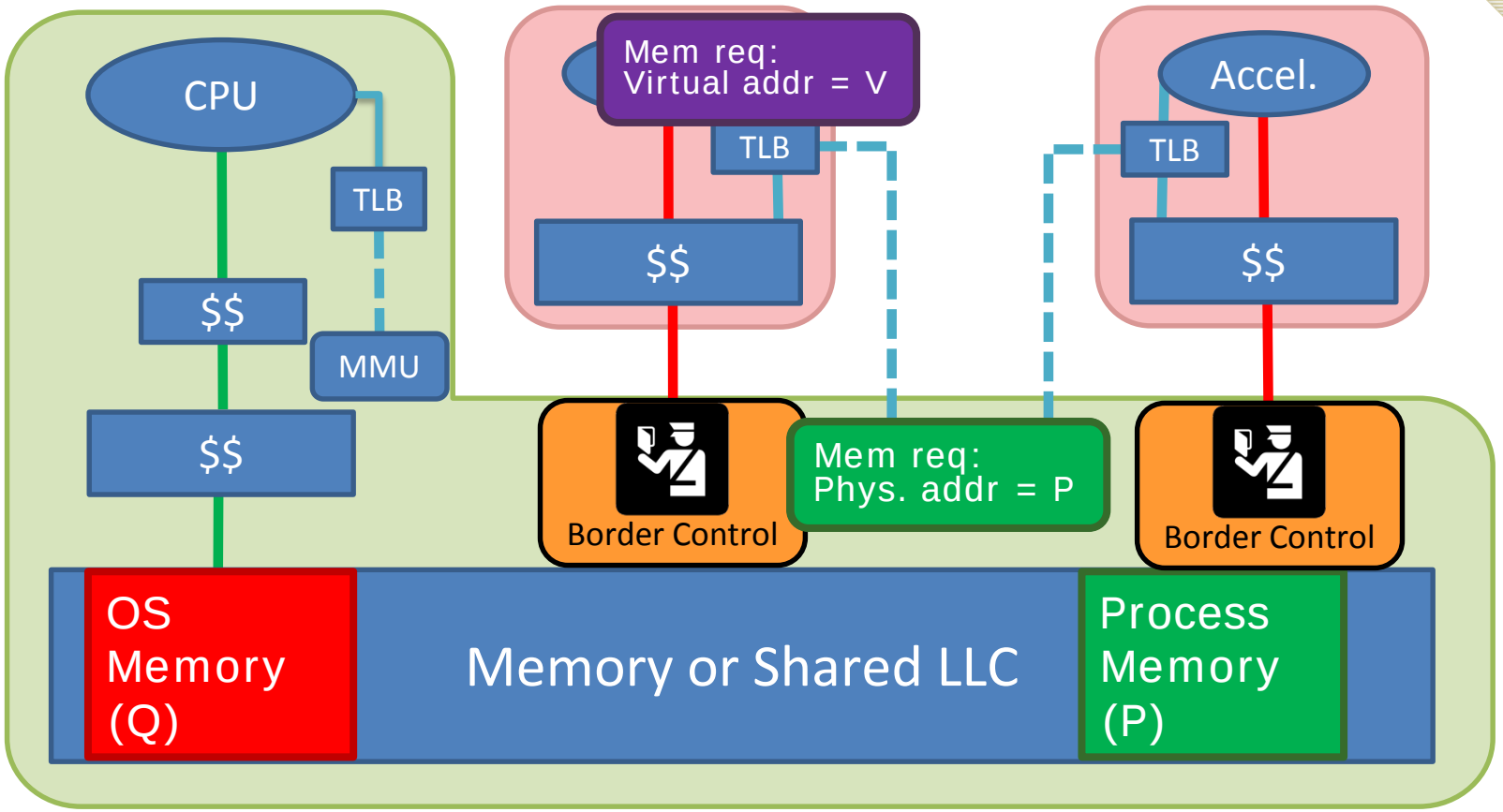
Trusted data path

Untrusted data path

Address translation path

Translation update path

Border Control



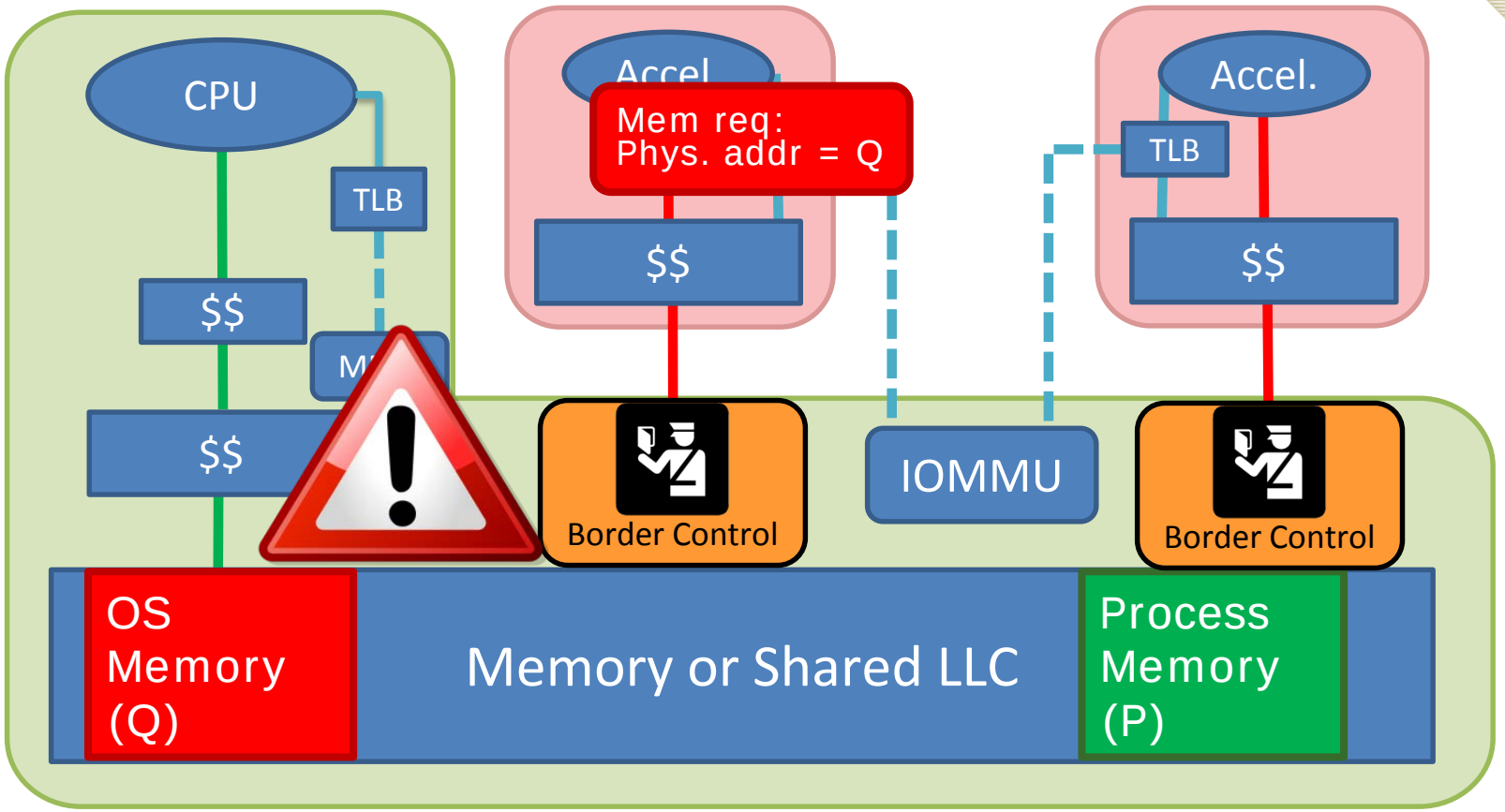
Trusted data path

Untrusted data path

Address translation path

Translation update path

Border Control



Trusted data path

Untrusted data path

Address translation path

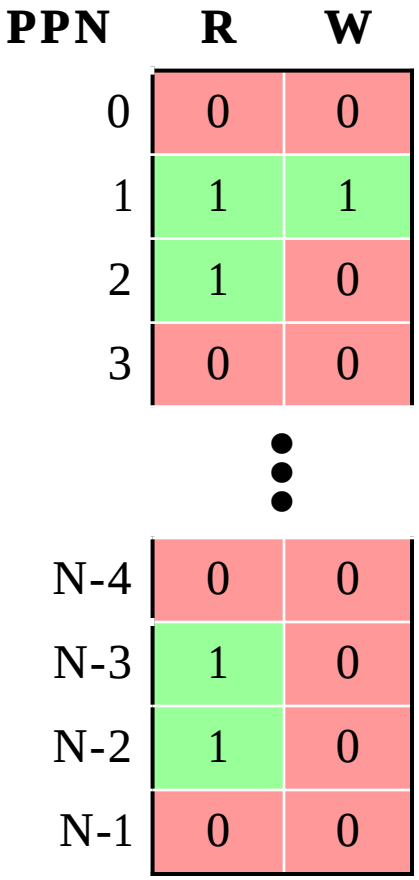
Translation update path

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

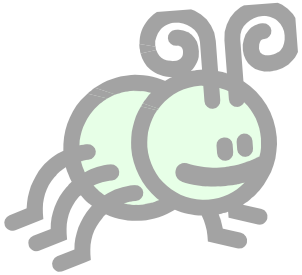


- 2 bits (R/W) per physical page
  - Initialized to 0 (no permission)
  - Lazily updated on IOMMU translation
  - Checked on all accelerator memory requests

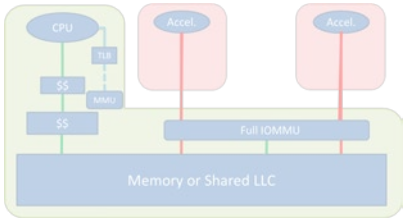
More details in paper!

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

Outline

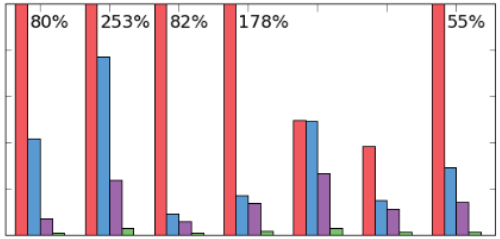


Motivation



Current Systems

Border Control

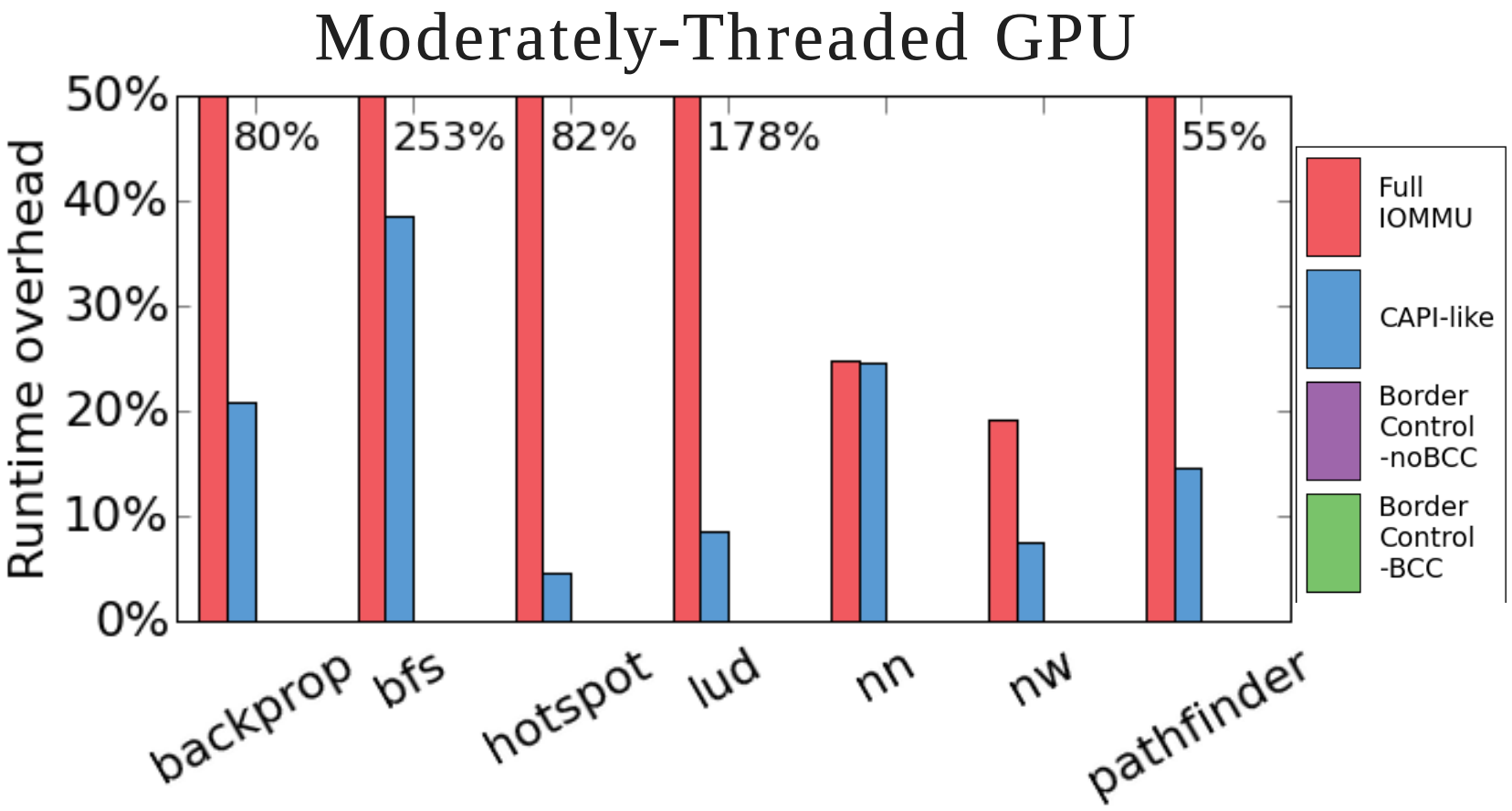


Evaluation

Methodology

- GPGPU → accelerator safety stress-test
- Simulator: gem5-gpu
  - Moderately-threaded: single core
  - Highly-threaded: eight cores
- Rodinia Benchmarks
- Baseline: fast but unsafe bypassable IOMMU

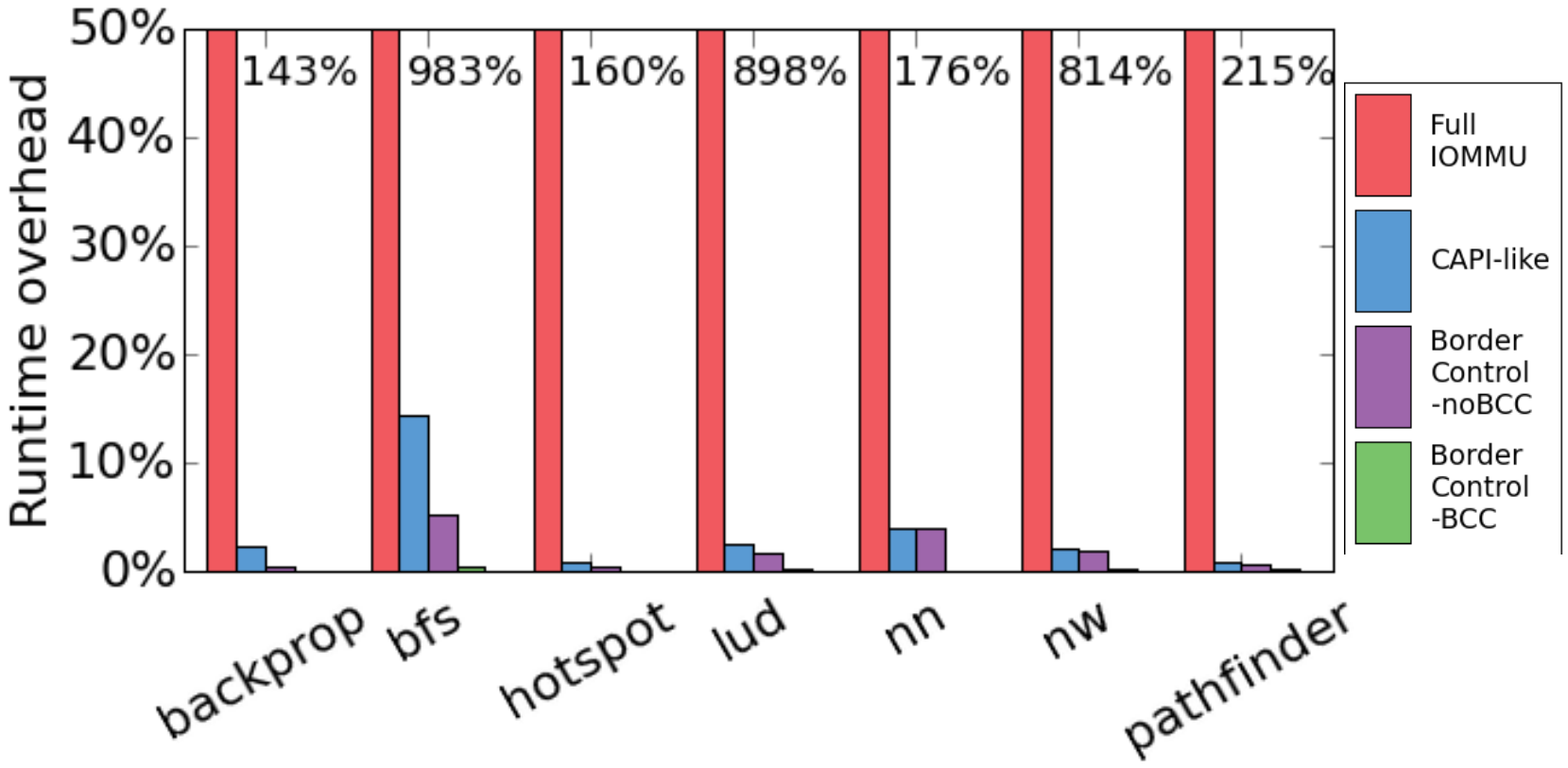
Border Control Overheads



Takeaway: Average **0.48%** performance overhead

Border Control Overheads

Highly-Threaded GPU



Takeaway: Average **0.15%** performance overhead

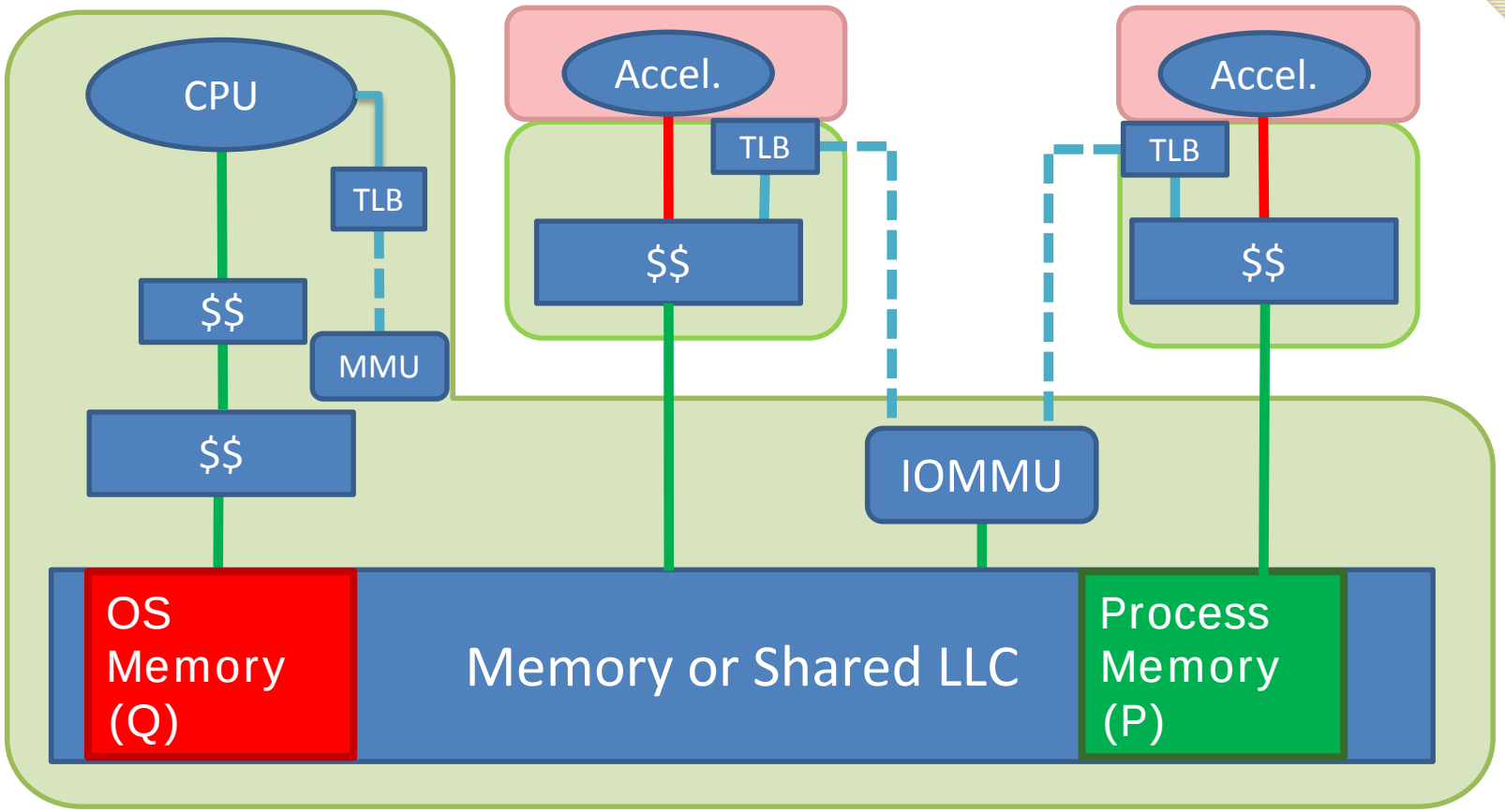
Conclusion

- Accelerators pose new security questions[1]
- 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



Trusted data path

Untrusted data path

Address translation path

Translation update path

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

CPU	
CPU Cores	1
CPU Caches	64KB L1, 2MB L2
CPU Frequency	3 GHz
GPU	
Cores (highly threaded) Cores (moderately threaded)	8 1
Caches (highly threaded) Caches (moderately threaded)	16KB L1, shared 256KB L2 16KB L1, shared 64KB L2
L1 TLB	64 entries
Shared L2 TLB (trusted)	512 entries
GPU Frequency	700 MHz
Memory System	
Peak Memory Bandwidth	180 GB/s
Border Control	
BCC Size	8KB
BCC Access Latency	10 cycles
Protection Table Size	196KB
Protection Table Access Latency	100 cycles

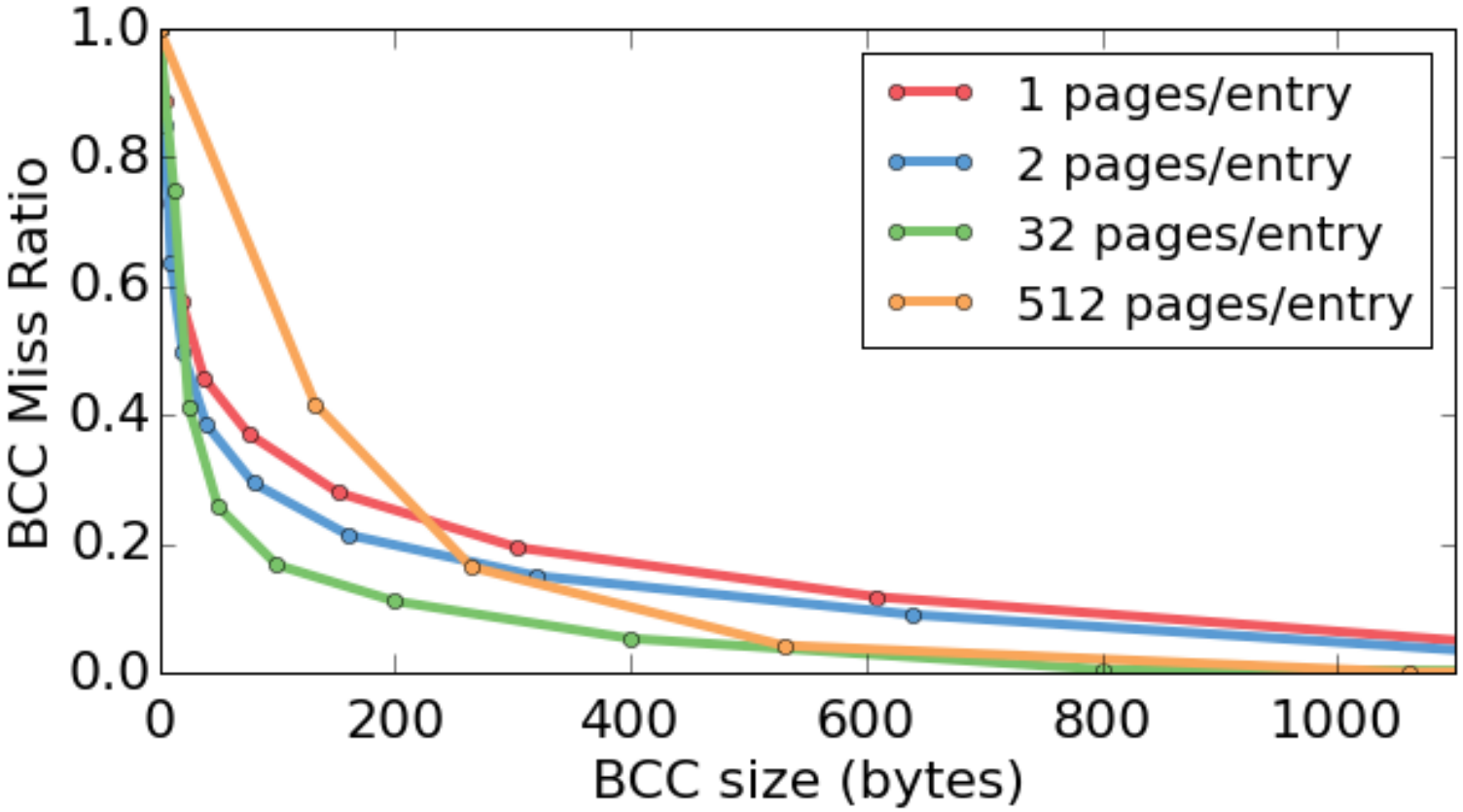
Table 3: Simulation configuration details.

Comparison of Configurations

	Safe?	L1 $	L1 TLB	L2 $	BCC
ATS-only IOMMU	✗	✓	✓	✓	N/A
Full IOMMU	✓	—	—	—	N/A
CAPI-like	✓	—	—	✓	N/A
Border Control-noBCC	✓	✓	✓	✓	—
Border Control-BCC	✓	✓	✓	✓	✓

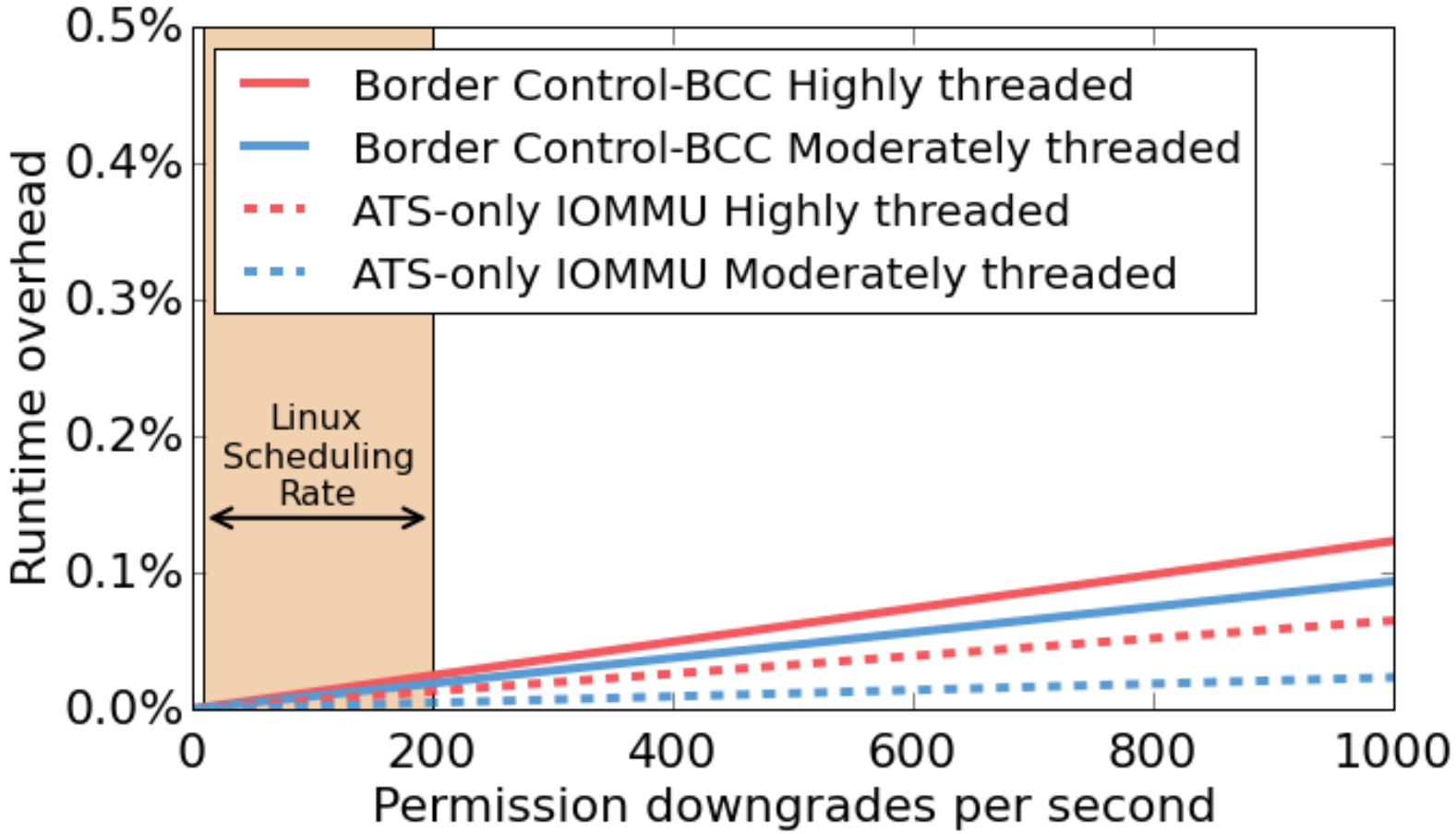
Table 2: Comparison of configurations under study.

Border Control Cache



Takeaway: A small (1KB) BCC is sufficient for our workloads

Border Control Flush Overhead



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