Practical DMA Attack Protection for FreeBSD

Brett Gutstein
Rice University
brett@gutste.in

BSDCam 2017
August 4, 2017

Motivations

DMA attack vulnerability

Exposed system buses

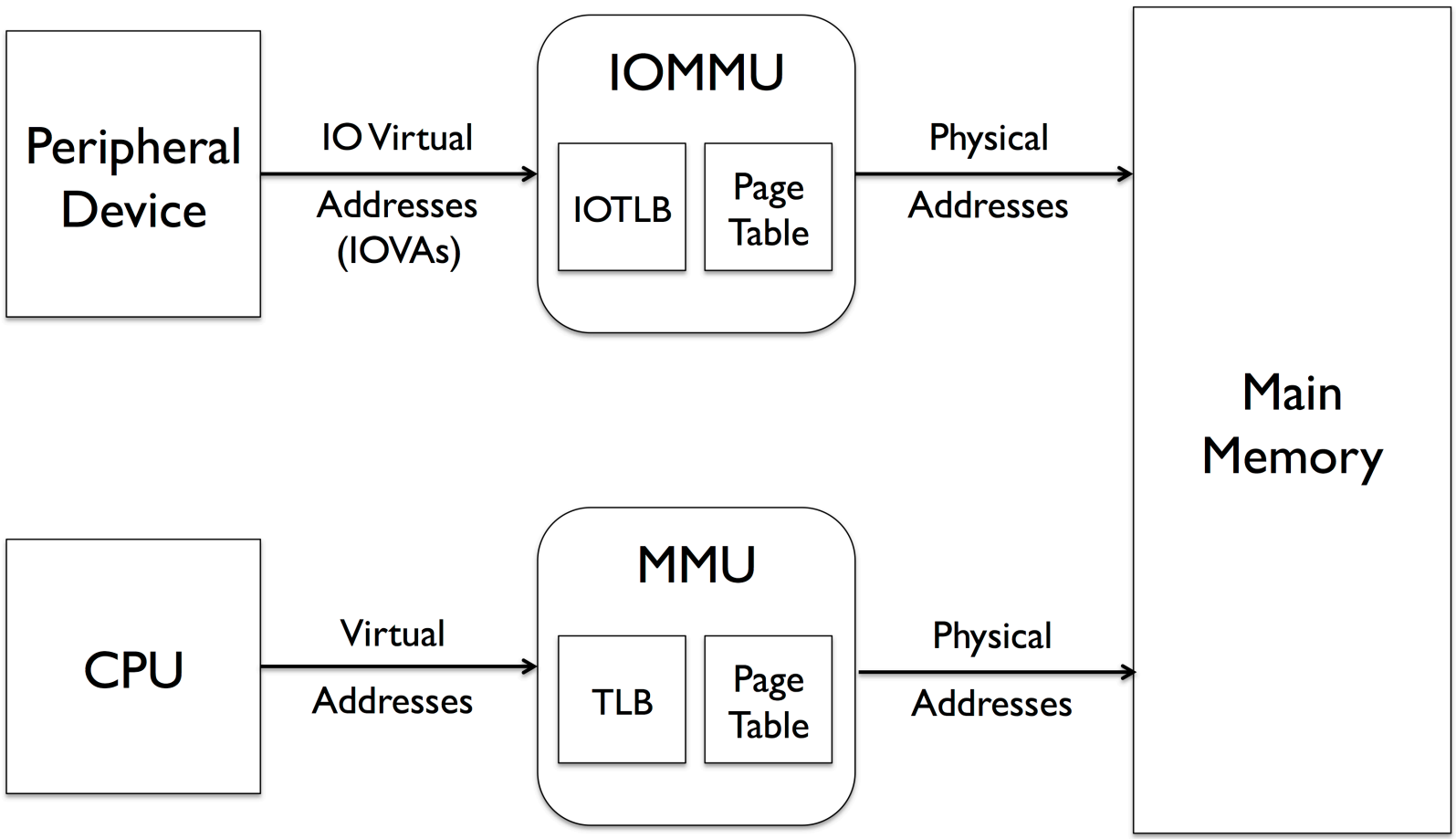
IOMMU driver overhead

Goals

Improve driver performance

Don’t sacrifice security properties

IOMMU


Peripheral Device
IO Virtual Addresses (IOVAs)
IOMMU
IOTLB
Page Table
Physical Addresses
Main Memory
CPU
Virtual Addresses
MMU
TLB
Page Table
Physical Addresses

VT-d Driver in FreeBSD

	IOVA Allocation	Page Table Synch.	Page Table Memory	IOTLB Invalidation
FreeBSD	Red-Black Tree	Coarse Lock	Reclaimed	Strict*

* loader tunable

Experimental Setup

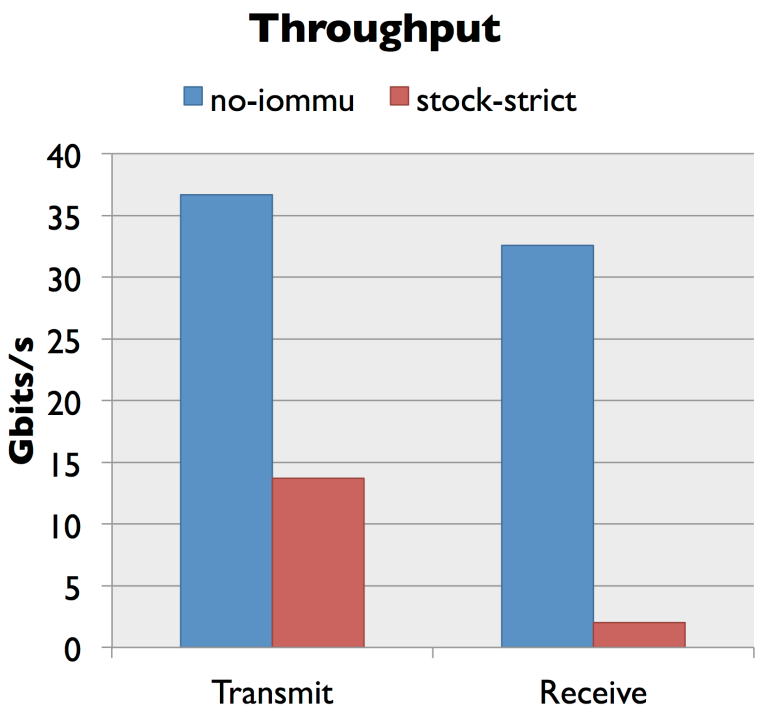
`iperf3` benchmark on 40G NICs

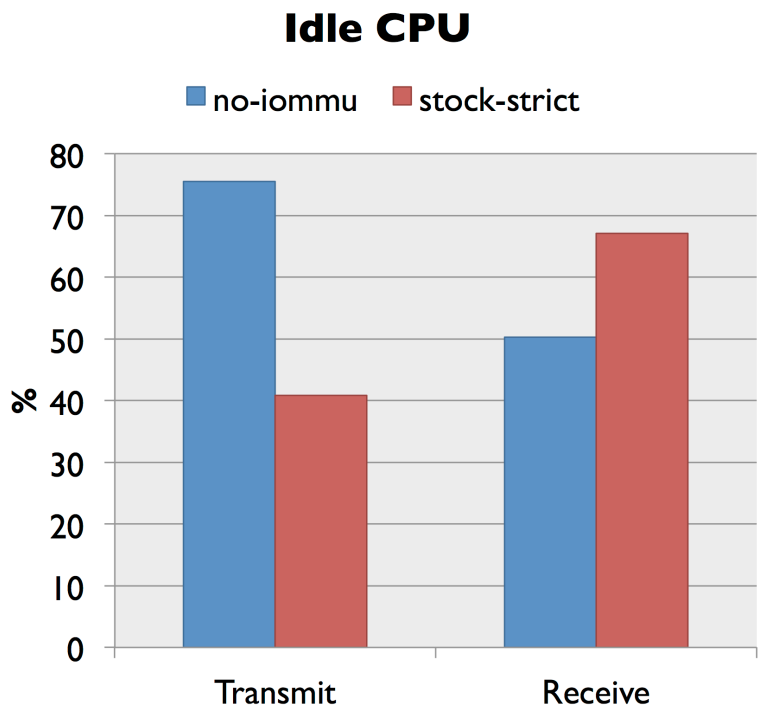
The machine under test has a 3.40 Ghz Intel Xeon E3-1231 Haswell CPU with 4 cores and hyperthreading disabled.

The supporting machine has a 3.50 Ghz Intel Xeon E3-1240 Skylake CPU with 4 cores and two hardware threads per core. It always operates with the IOMMU disabled.

Both machines have 32GB of DDR3 RAM and are connected back-to-back by Intel XL710 QDA1 40G NICs. They run FreeBSD HEAD with debugging kernel options disabled.

Baseline Performance





A New Design

Idea: have the IOVA allocator handle page table management

Vmem hierarchy:

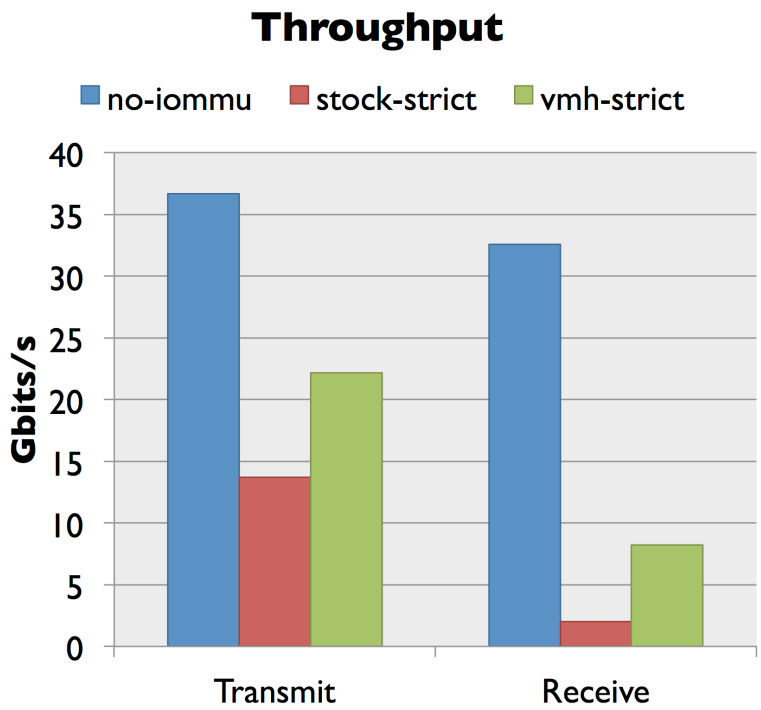
- constant-time IOVA allocation
- implicit page table synchronization
- per-CPU caching

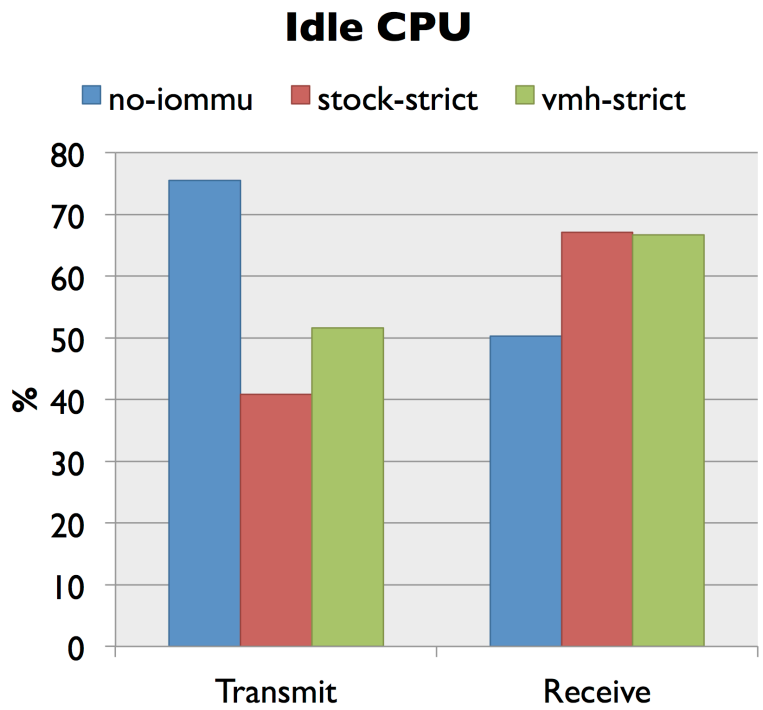
VT-d Driver Comparison

	IOVA Allocation	Page Table Synch.	Page Table Memory	IOTLB Invalidation
BSD-vmh	Vmem Hierarchy (CPU Caches)	Implicit, Lock Free	Reclaimed	Strict*
FreeBSD	Red-Black Tree	Coarse Lock	Reclaimed	Strict*

* loader tunable

Results


Throughput
no-iommu
stock-strict
vmh-strict
Gbits/s
40
35
30
25
20
15
10
5
0
Transmit
Receive

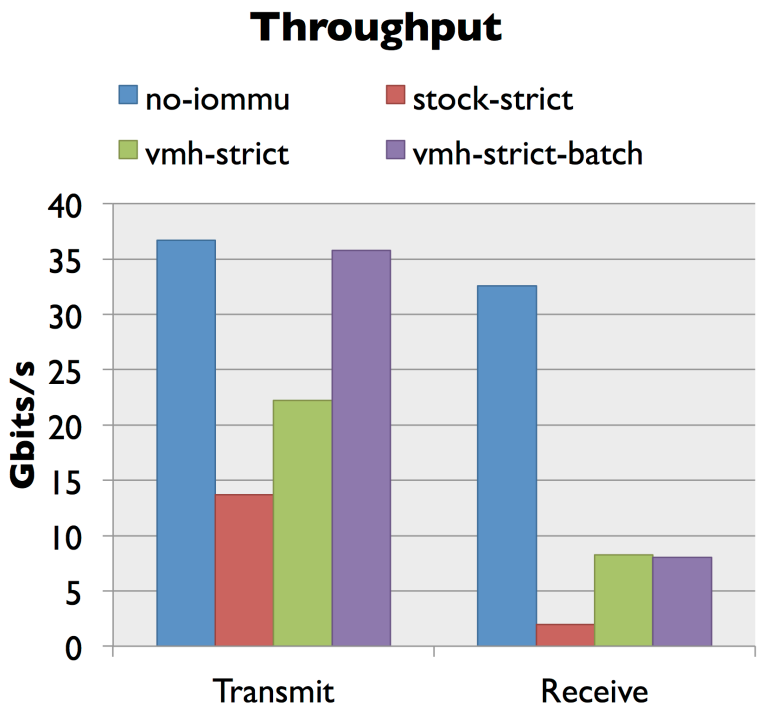

Idle CPU
no-iommu
stock-strict
vmh-strict
%
80
70
60
50
40
30
20
10
0
Transmit
Receive

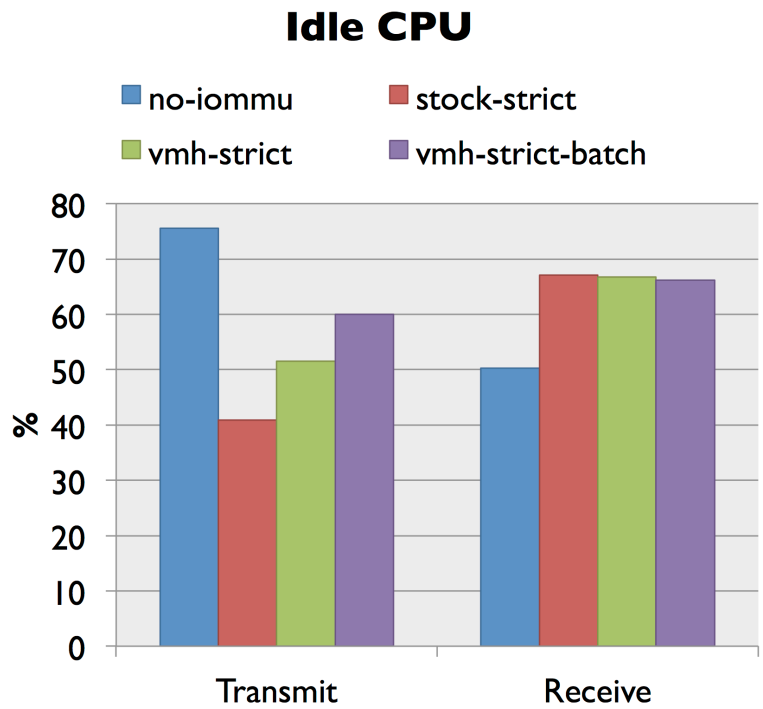
Batching in BUS_DMA(9)

`bus_dmamap_load_mbuf_sg()` made a call to the IOMMU driver for each **`mbuf`** in a chain

Solution: preprocess **`mbuf`** chains and make a single call

Results


Throughput
no-iommu
stock-strict
vmh-strict
vmh-strict-batch
Gbits/s
40
35
30
25
20
15
10
5
0
Transmit
Receive


Idle CPU
no-iommu
stock-strict
vmh-strict
vmh-strict-batch
%
80
70
60
50
40
30
20
10
0
Transmit
Receive

Future Work

Optimize additional IO paths

Develop a generic IOMMU library

Add security properties