Memory Protection in Kernel Space

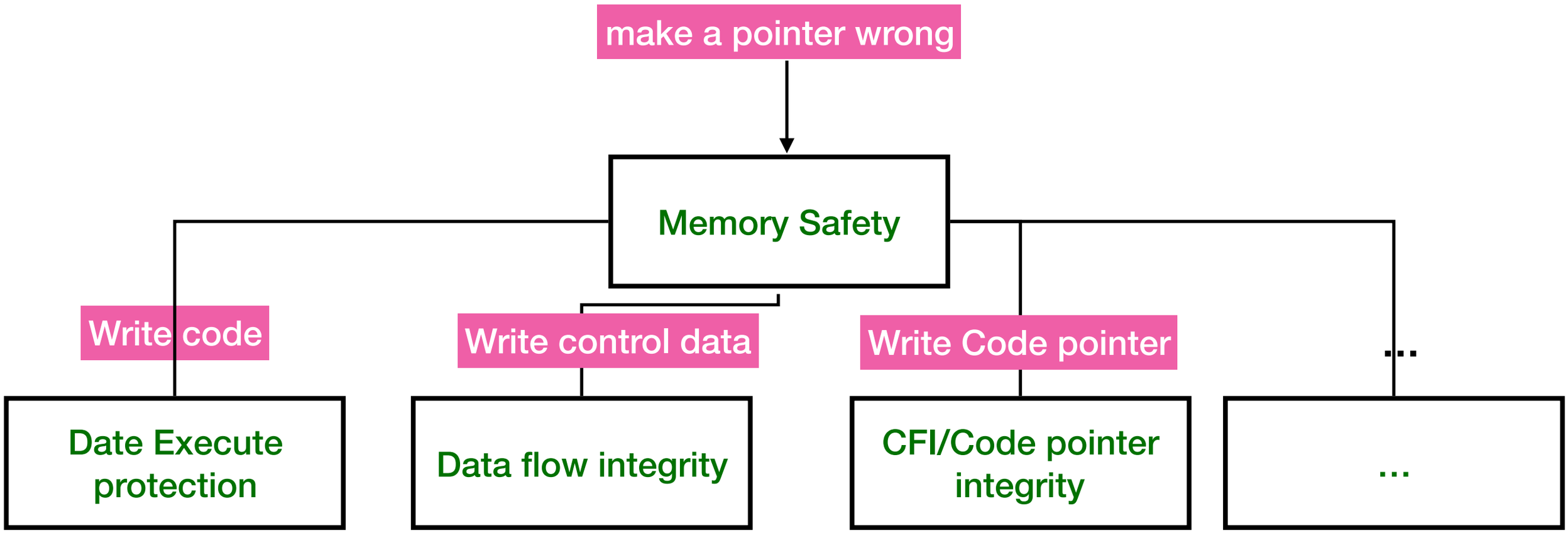
What’s wrong with kernel?

- Most of kernel written in C -> lead to memory buggy
- Faulty: a bug happens in driver may halts whole kernel
- Security: a vulnerability in driver may break hurts whole kernel.

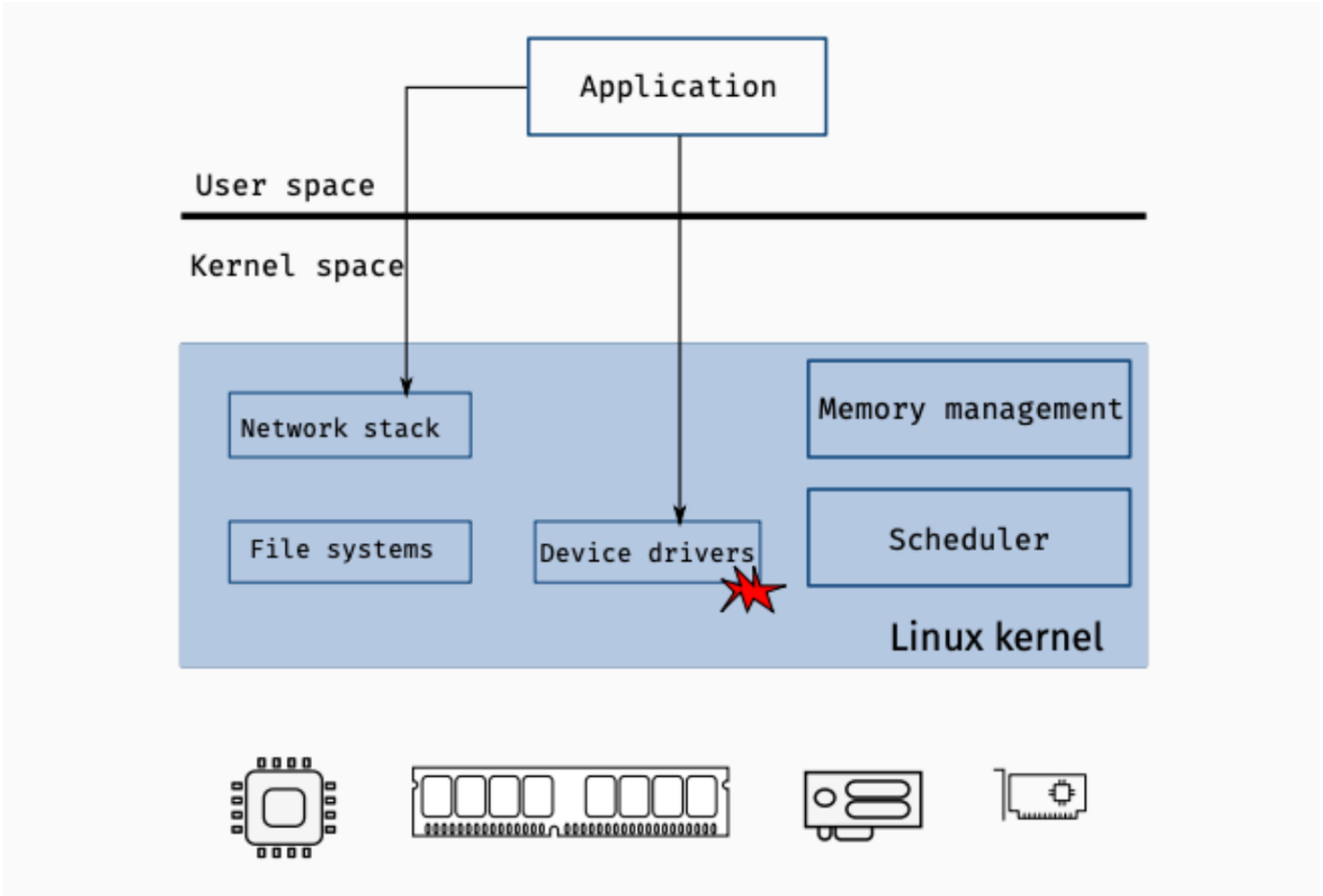
Mitigate Memory Bug(Or its influence)

- Code pointer integrity
- Kernel data flow integrity
- KASLR
- DEP(KR^2)
- None executable to user code in privileged level.

Memory bugs are hard to eliminate



Monolithic kernel propagates bugs



Isolation in monolithic kernel

Hardware based

Heiser **SUD** **VirtuOS**

Software based

BGI **XFI** **LXFI**

Microdriver/Decaf

Rust driver(hotos19)

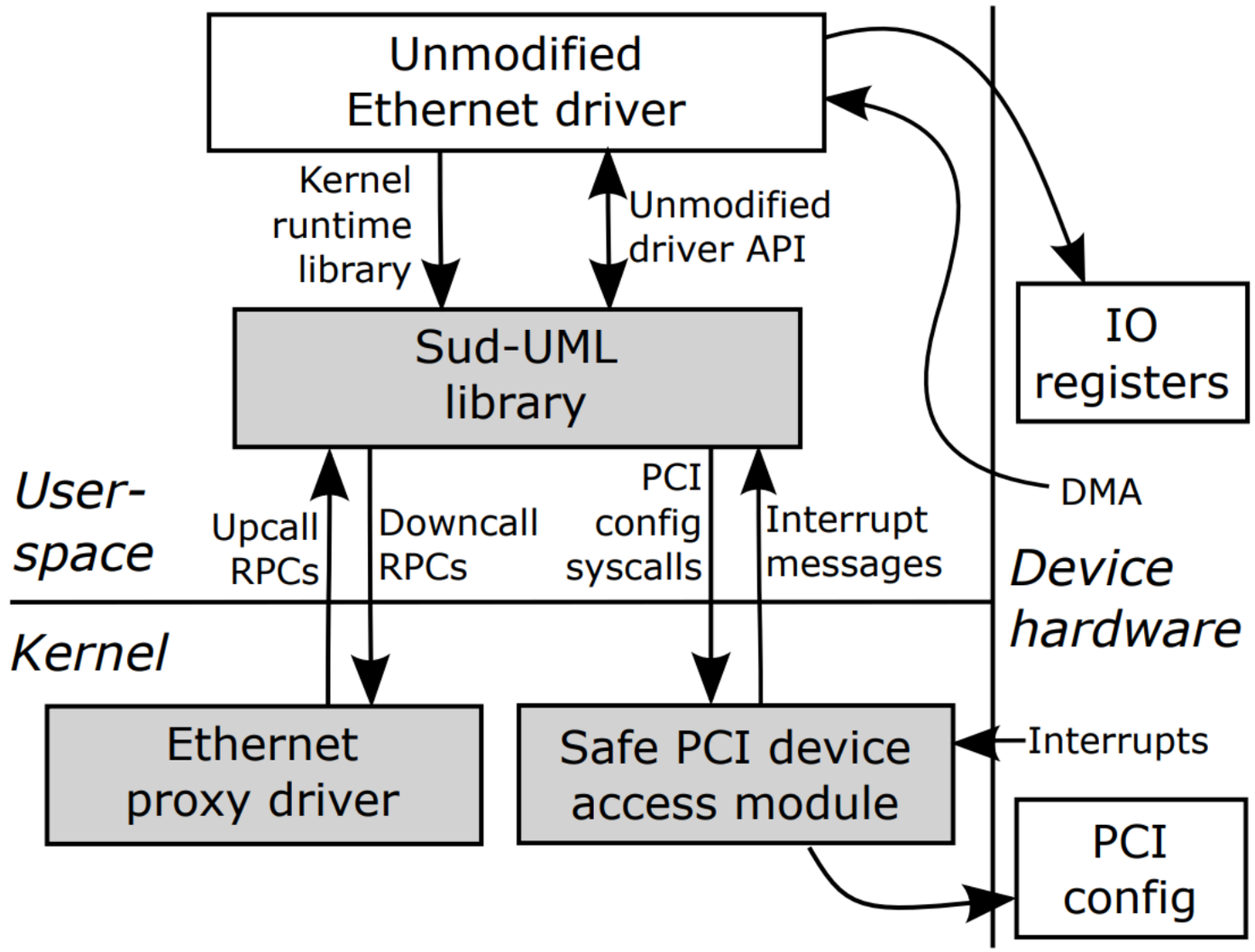
Develop effort?

Performance?

Address space isolation

- Provide kernel driver execute environment
- Driver need to access kernel data.
- Context switch

SUD


Unmodified
Ethernet driver
Kernel
runtime
library
Unmodified
driver API
Sud-UML
library
IO
registers
DMA
User-
space
Upcall
RPCs
Downcall
RPCs
PCI
config
syscalls
Interrupt
messages
Device
hardware
Kernel
Ethernet
proxy driver
Safe PCI device
access module
Interrupts
PCI
config

SUD

- Develop effort
  - User mode Linux(UML), proxy driver
- Performance: context switch

Table 1: Latency of Basic Operations

Instruction or Operation	Cycles*
write to `CR3` with `CR3_NOFLUSH`	186 ± 9
`vmfunc`	109 ± 1
`wrpkru`	26 ± 2
no-op system call w/ KPTI	433 ± 12
no-op system call w/o KPTI	96 ± 2
no-op VM call	1694 ± 131
user-space context switch	748 ± 8
process context switch using semaphore	4426 ± 41

* ± half the width of the 95% confidence interval

Use EPT:

- Context switch between user and kernel are too heavy
- Use EPT(VirtuOS)?

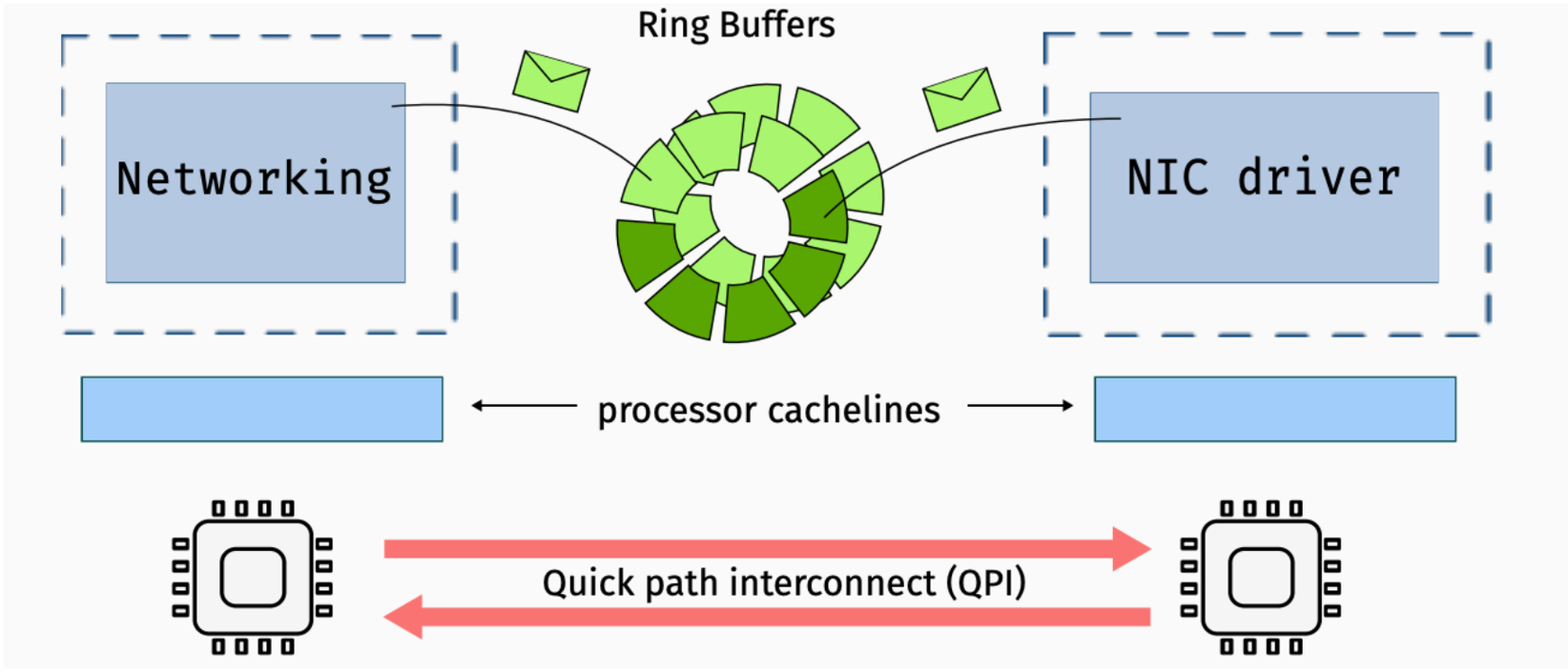
Use EPT:

- Context switch between user and kernel are too heavy
- Use EPT(VirtuOS)?
- NO, it’s still too heavy to switch context:
- LXDs: use async queue(cache line sync)

Operation	Cycles
seL4 same-core d820 (without PCIDs)	1005
seL4 same-core c220g2 (with PCIDs)	834
LXDs cross-core r320 (non-NUMA)	448
LXDs cross-core d820 (NUMA)	533

Table 2: Intra-core vs cross-core IPC.

LXDs



- Do not switch context(driver threads always resides in)
- Wait others to feed from last level cache

LXDs

- Performance test on 8 core machine:

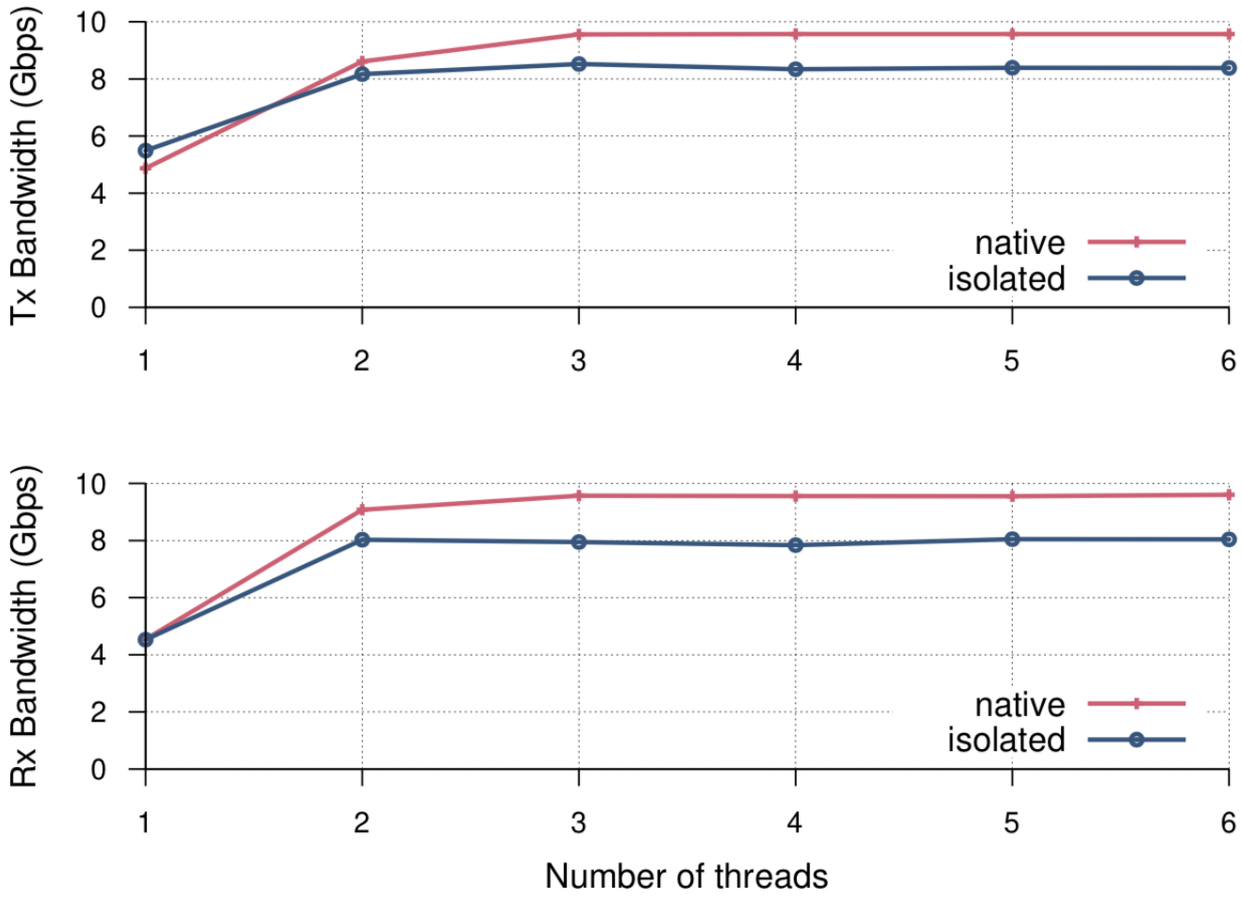


Figure 4: Ixgbe Tx and Rx bandwidth.

LXDs

- Performance test on 8 core machine:

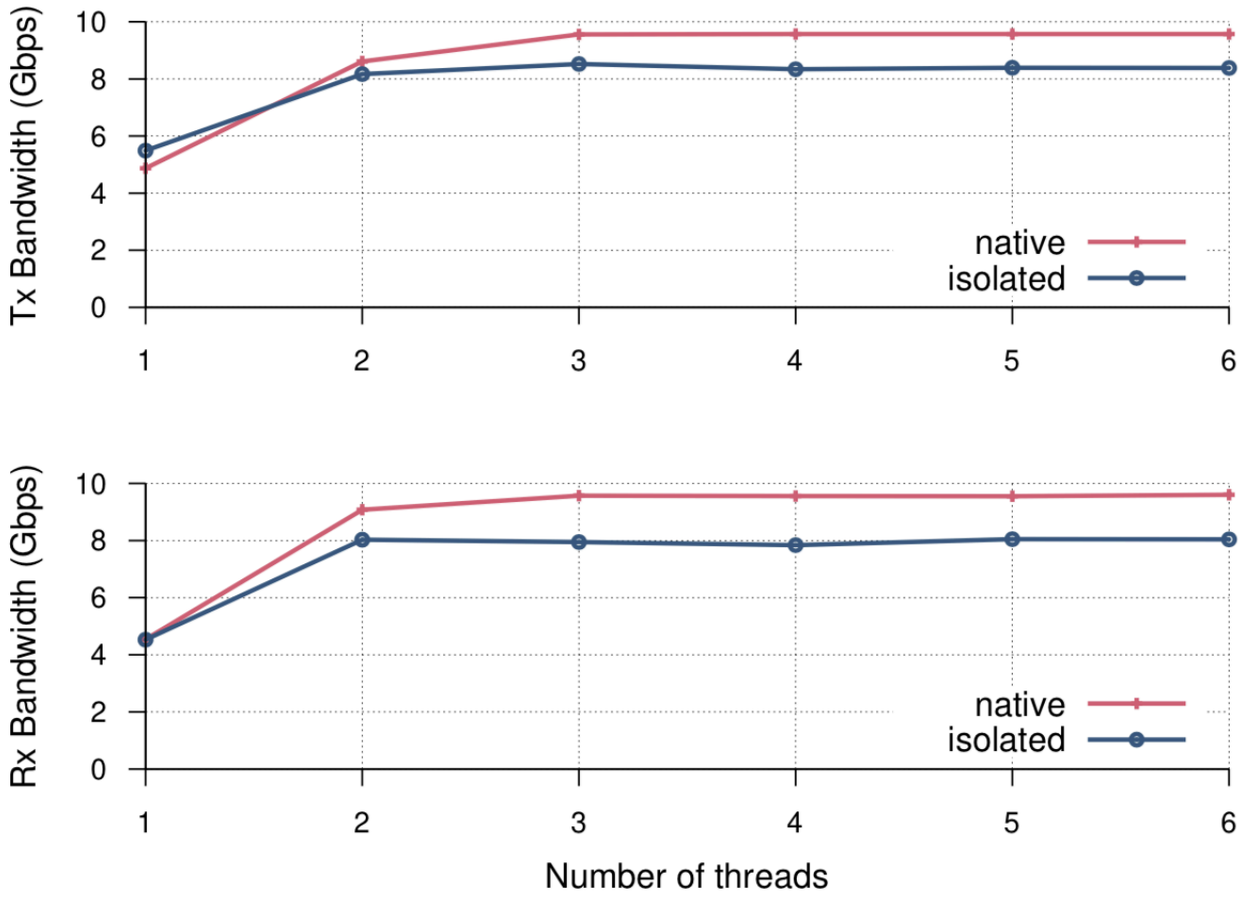


Figure 4: Ixgbe Tx and Rx bandwidth.

Q: why up to 6 thread on 8 core machine?

LXDs: cons

- Burn extra cores
- ASYNC are not suitable to complex driver(like usb driver)
- EPT only count the direct cost: nested virtualization needed.

Any other good way?

- Single address space isolation:
- Domain page model

Domain page model

- Protection domain = accessible pages + page permission
- Each page is belongs to a domain.

	Page 1	Page 2	Page 3	Page 4	Page 5	Page 6
Domain A	R		RW		RWX	
Domain B		R-X		R	R	R
Domain C	RWX	R-X	RW			

State-of art protection mechanism

- ARM domain access control register
- Intel memory protection keys

State-of art protection mechanism

- ARM domain access control register (DACR)

- Each page entry has a 4 bits fields, indicate the domain it belongs. Totally 2^4 = 16 domains.

- DACR (32 bits) has two bits for each domain, means currently the cpu can/cannot access which domain.

b00 = No access. Any access generates a domain fault.

b01 = Client. Accesses are checked against the access permission bits in the TLB entry.

b10 = Reserved. Any access generates a domain fault.

b11 = Manager. Accesses are not checked against the access permission bits in the TLB entry, so a permission fault cannot be generated. Attempting to execute code in a page that has the TLB *eXecute Never* (XN) attribute set does not generate an abort.

DACR usage currently

- User level sandbox: shred and ARMlock
- Save page table for shared libraries on Android:
- Kernel drivers: DIKernel

Limitions

- Only 16 domains support, easy to swap in 64 bits page table but…
- DACR only on arm 32, while MPK only for user space.

Still thinking…

- Why write to CR3 are slower than vmfunc?
- Address space isolation: more than direct cost.
- Similar problem: speed up IPC in microkernel(Skybridge)