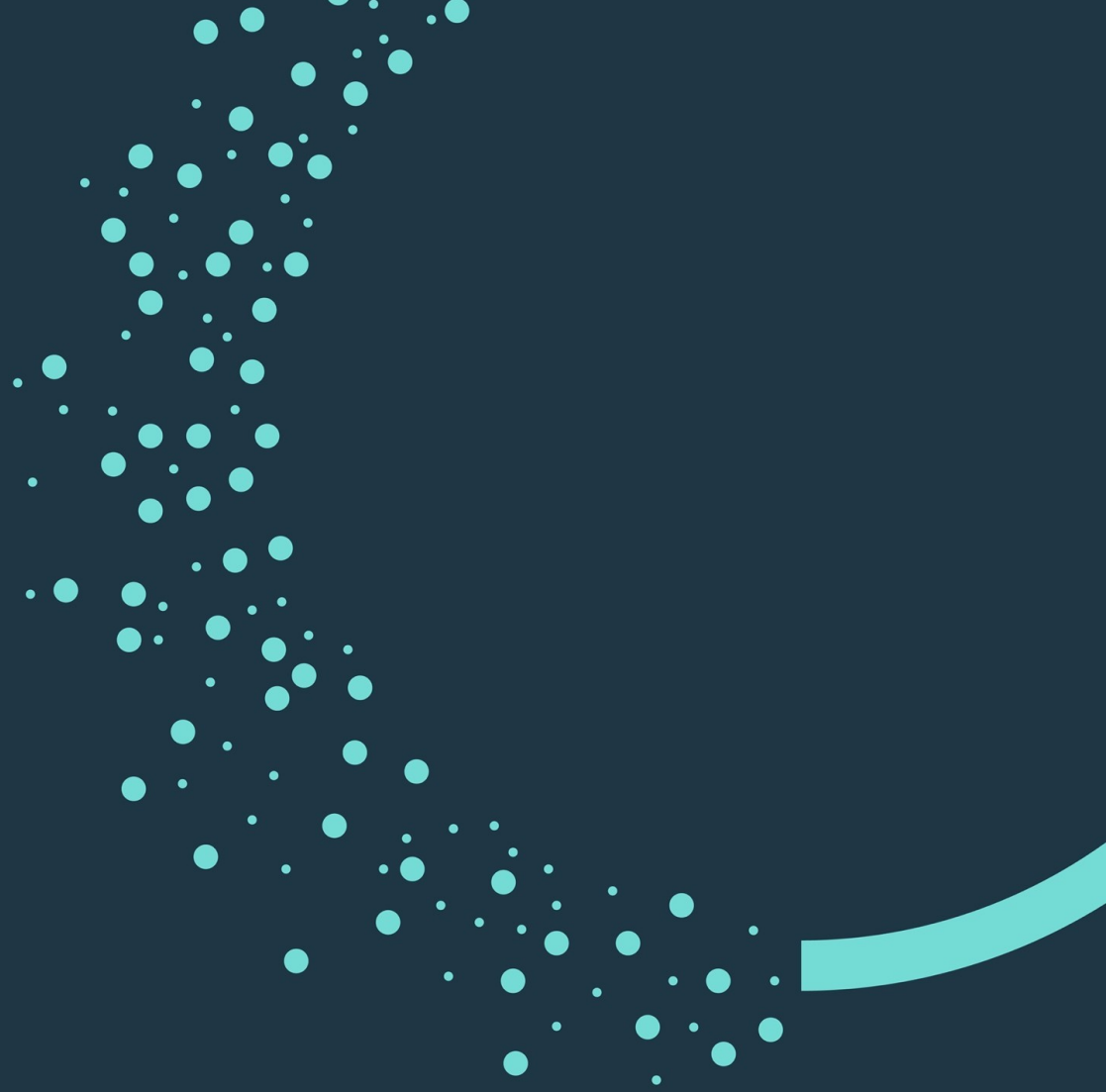


seL4 VMs with virtio à la KVM/QEMU

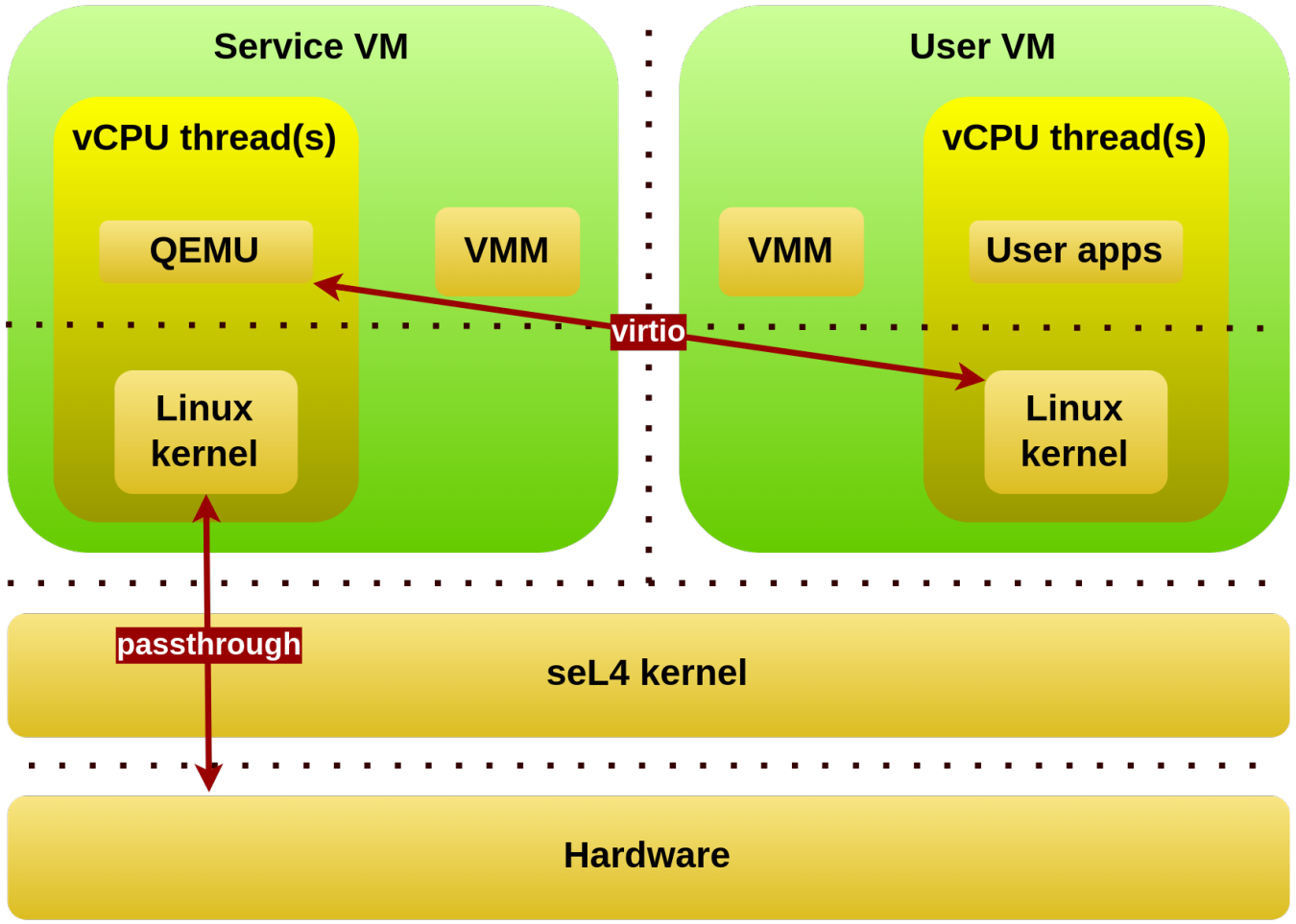
Hannu Lyytinen, Markku Ahvenjärvi
seL4 summit 2022 – Munich, Germany

Secure Systems
Research Center

Why?

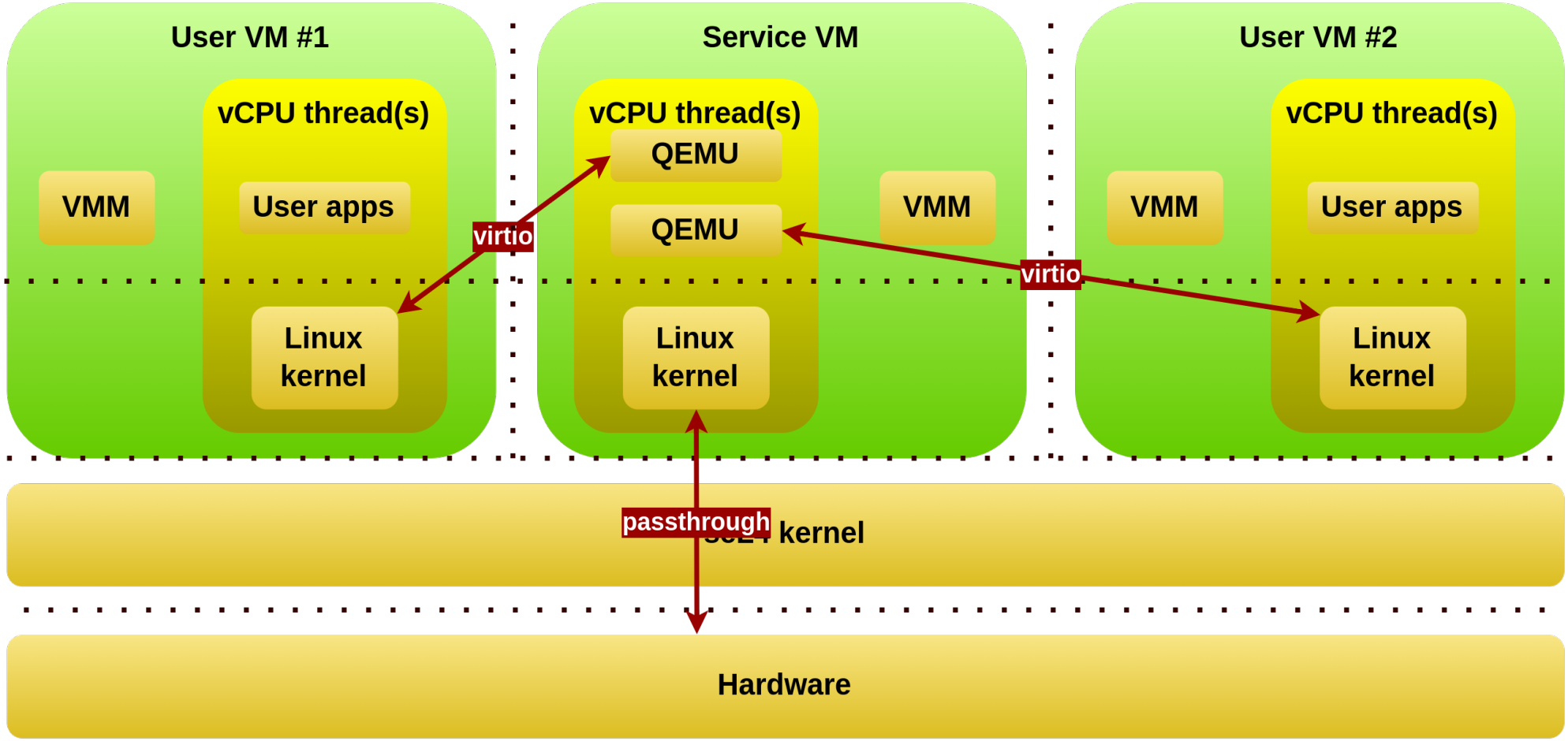
- Internal virtualization projects which use KVM
- Type-1 hypervisor better, no huge TCB
- Could we replace KVM with seL4?
- How to bring QEMU's (KVM's "little helper") virtio to seL4?
- Could we use CrosVM instead of QEMU (virtio-wayland)?

Our seL4 idea



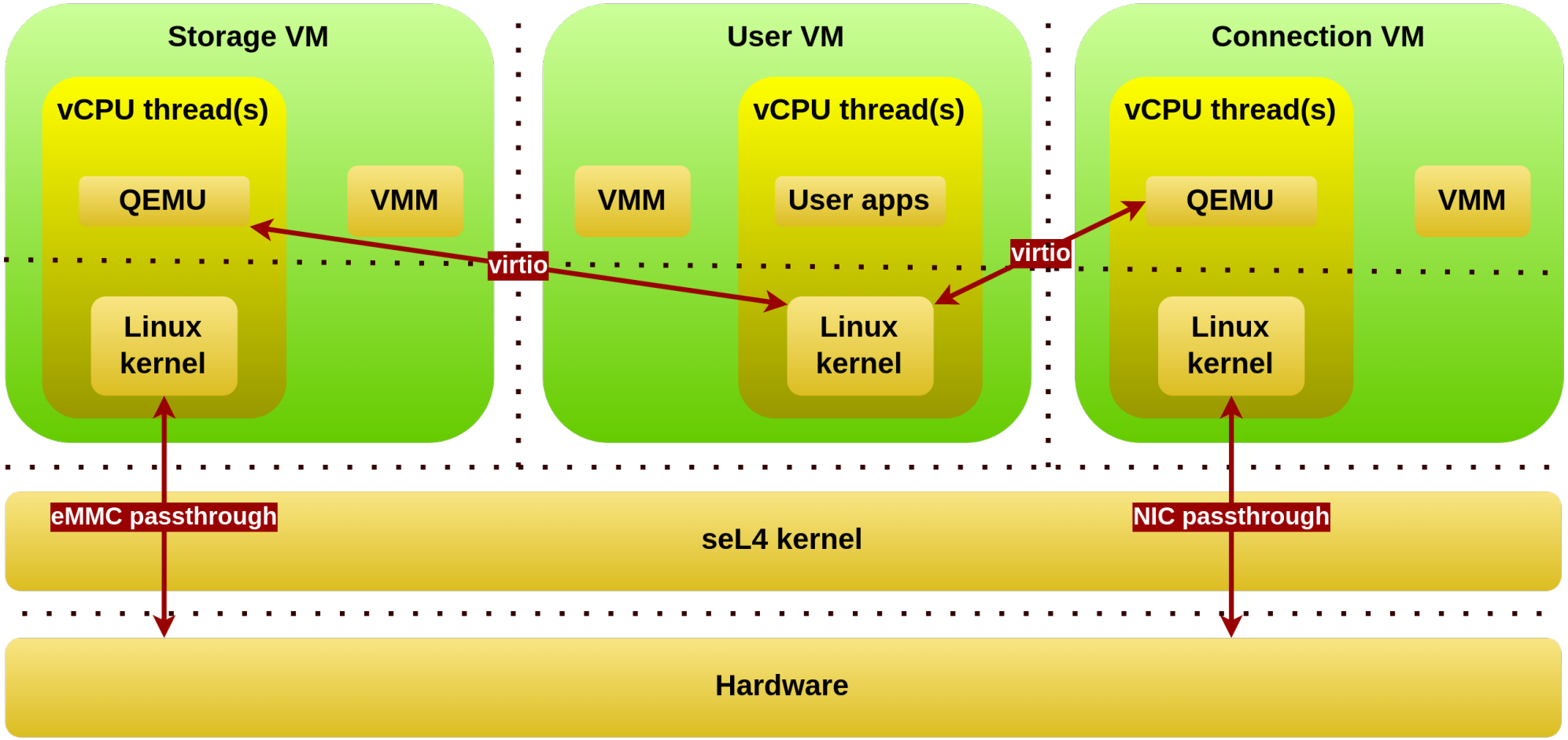
Secure Systems Research Centre

Our seL4 idea



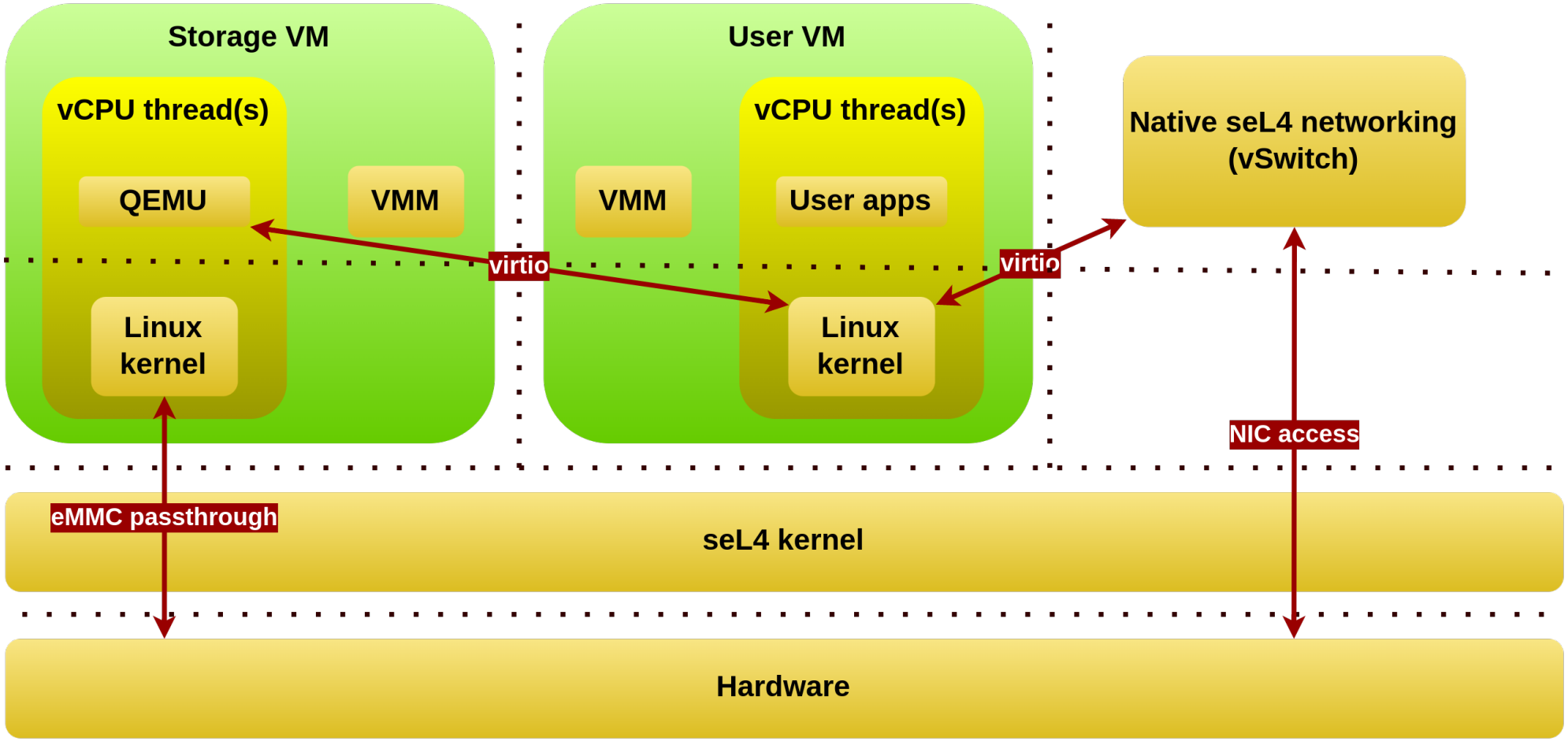
Secure Systems Research Centre

Multihost



Secure Systems Research Centre

Mixed VM/native



Secure Systems Research Centre

Where are we now

- “modern” virtio 1.0 supported
- virtio-console, virtio-blk, virtio-net, virtfs
- virtio-display, virtio-gpu, virtio-input (mouse, keyboard)
- PCI transport; no need to configure anything in guest
- Some preliminary results
- rpi4 virtio-blk 70 MB/s
- rpi4 virtio-net 1 MB/s (3% of native, ouch)
- rpi4 virtio-gpu glx-gears 45 fps @ 720p
- One service VM / one user VM
- Host sees all of guest’s memory

Secure Systems Research Centre

What we are working on

- System-wide ftrace (kernel, VMMs, guests)
- Traces absolutely required to understand all interactions
- vhost should help with inferior virtio-net performance
- Minimize context switches
- Hypervisor/guest scheduler options study
- 1:1 service/user → M:N service/user
- Restrict guest memory visibility to host
- Dynamic VMs (pure virtio, no need for IOMMU/SMMU)
- CrosVM and its derivatives

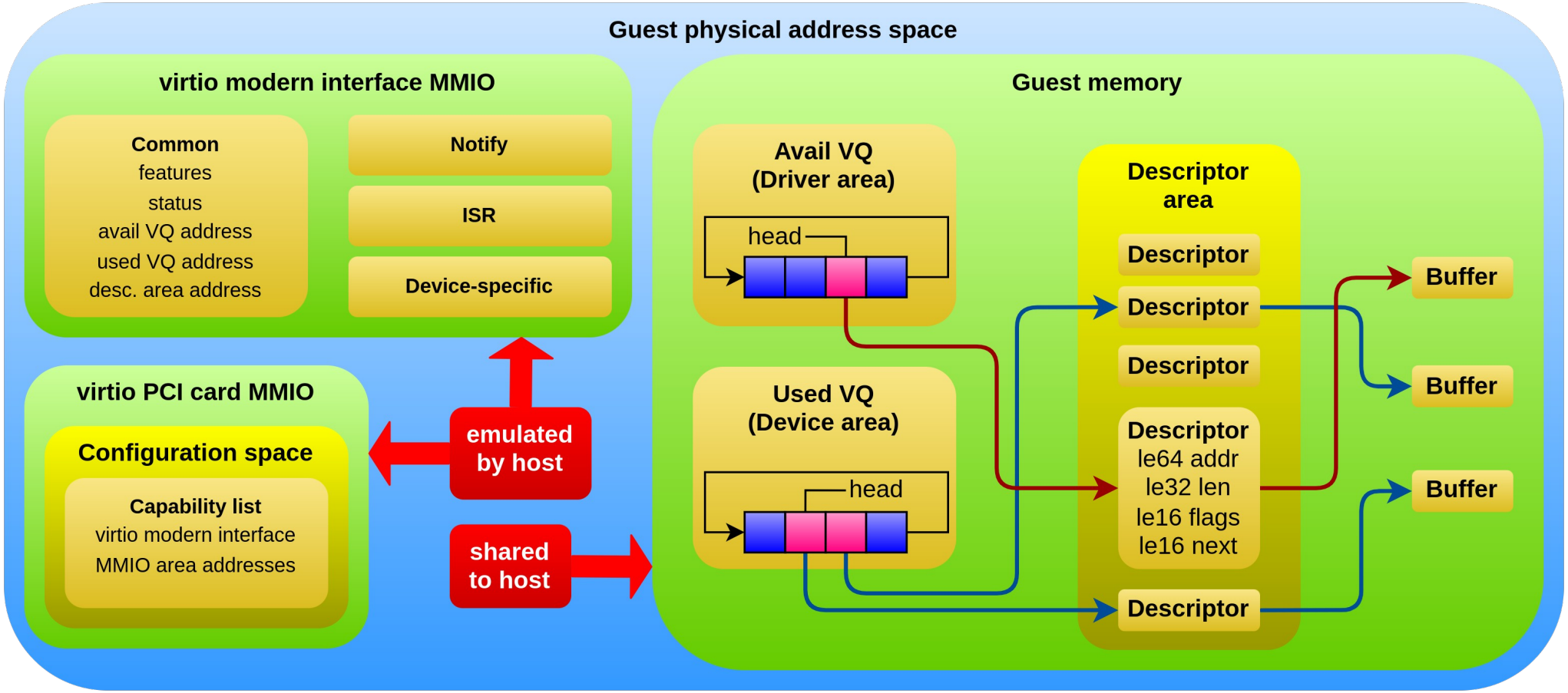
Secure Systems Research Centre

VMs with RPi4

- Docker container to build it all
- seL4 kernel, vm_qemu_virtio example, guest Yocto Linux root filesystems
- Boot from network with DHCP / TFTP
- First VM uses NIC passthrough and mounts root from NFS
- Second VM mounts root from virtio-blk provided by QEMU in first VM
- So effectively both root filesystems served from development desktop
- Quick iteration cycle, no need to play with SD cards
- ”screen” to multiplex VM consoles onto RPi4’s physical UART
- QEMU’s debug monitor also in the screen session
- Try it out: https://github.com/tiiuae/tii_sel4_build

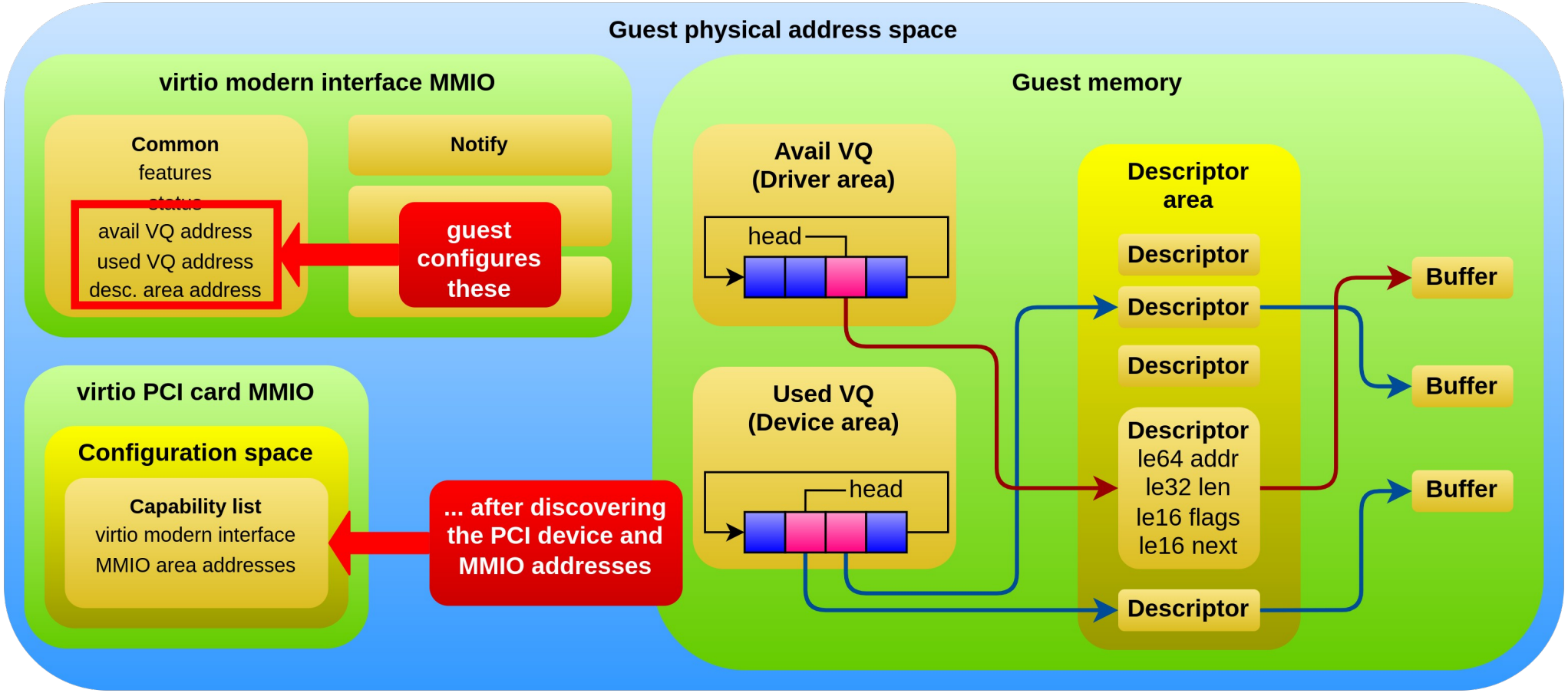
Secure Systems Research Centre

Virtio recap



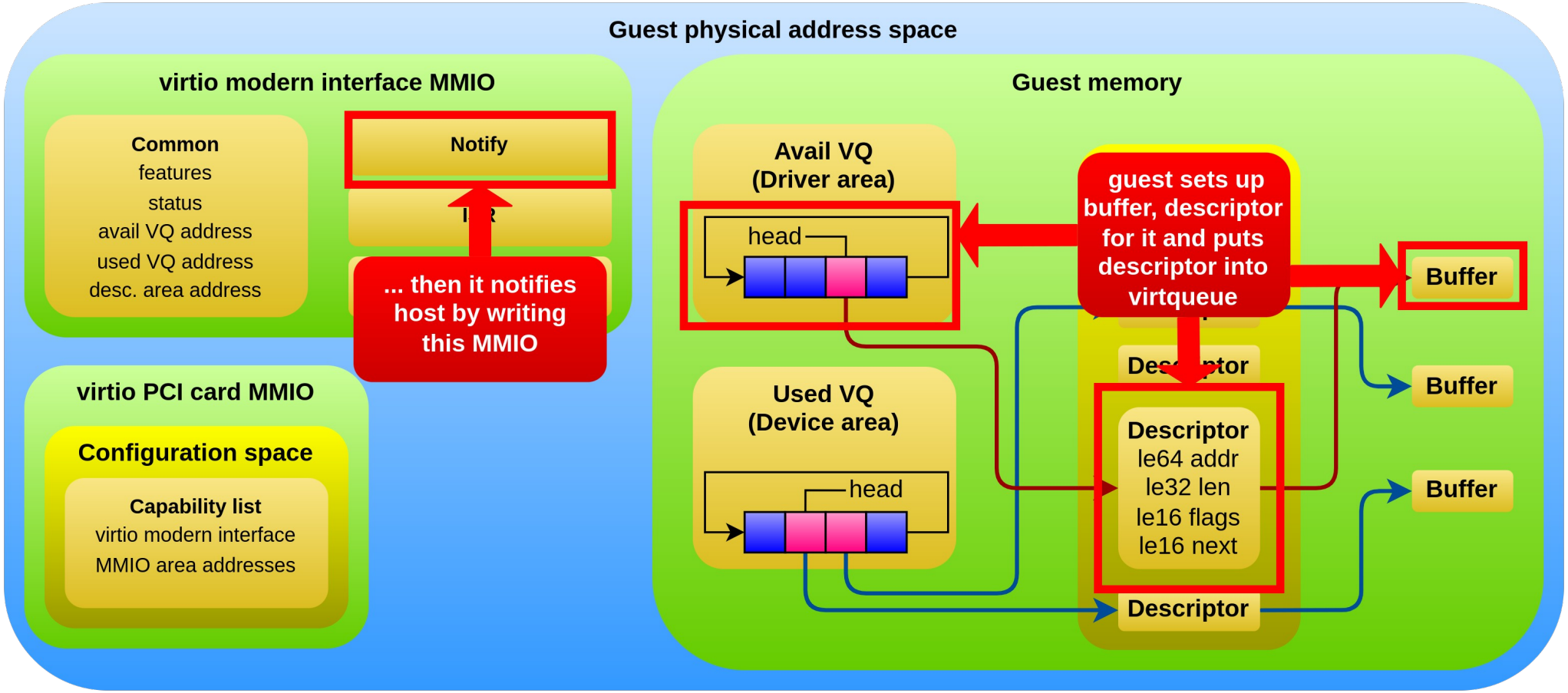
Secure Systems Research Centre

Virtio recap



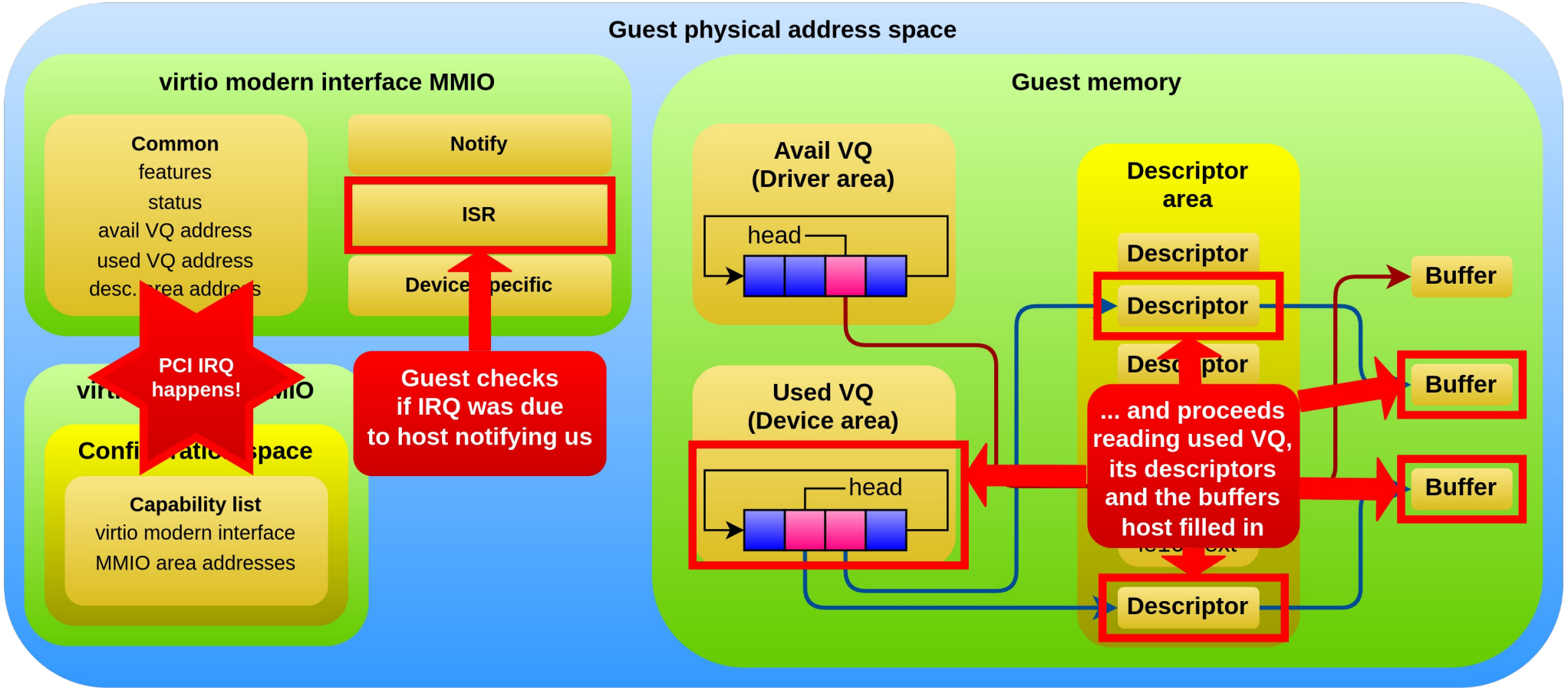
Secure Systems Research Centre

Virtio recap



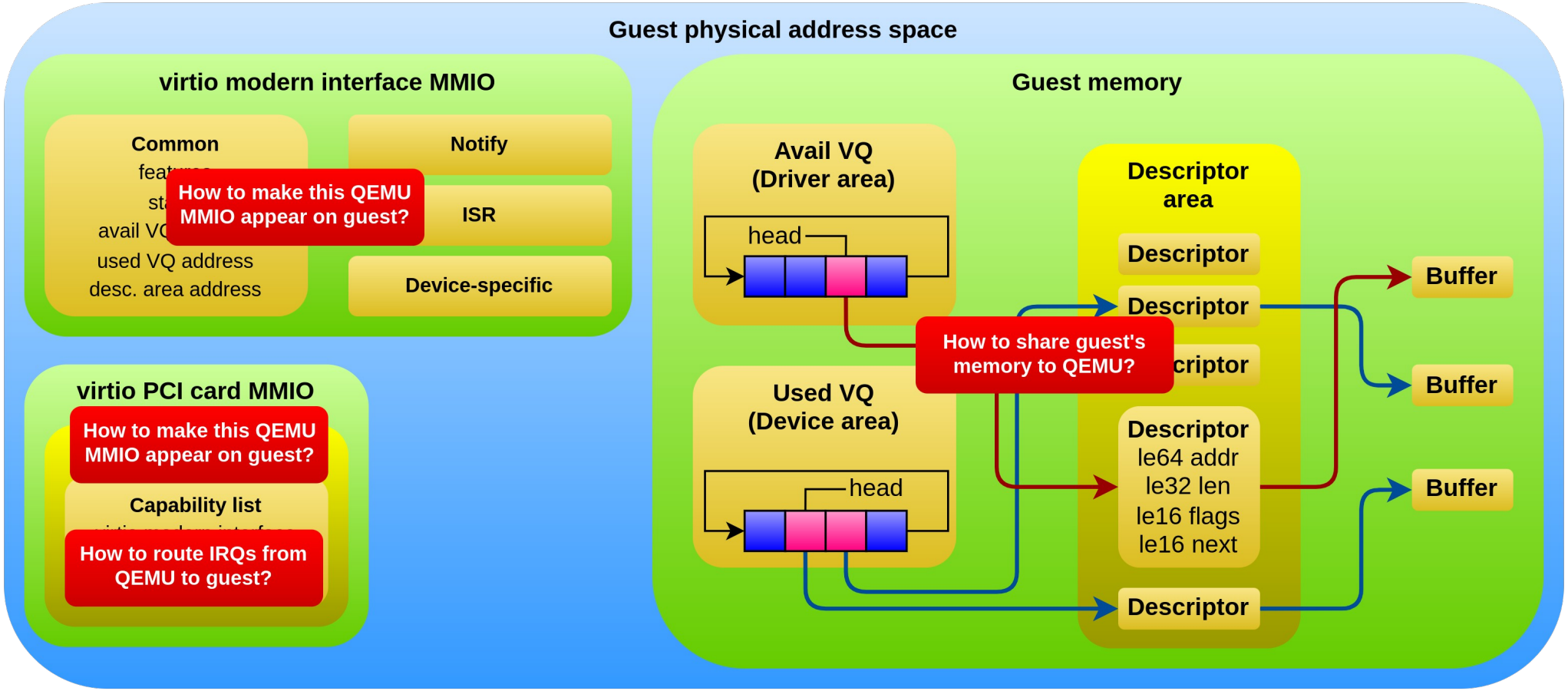
Secure Systems Research Centre

Virtio recap



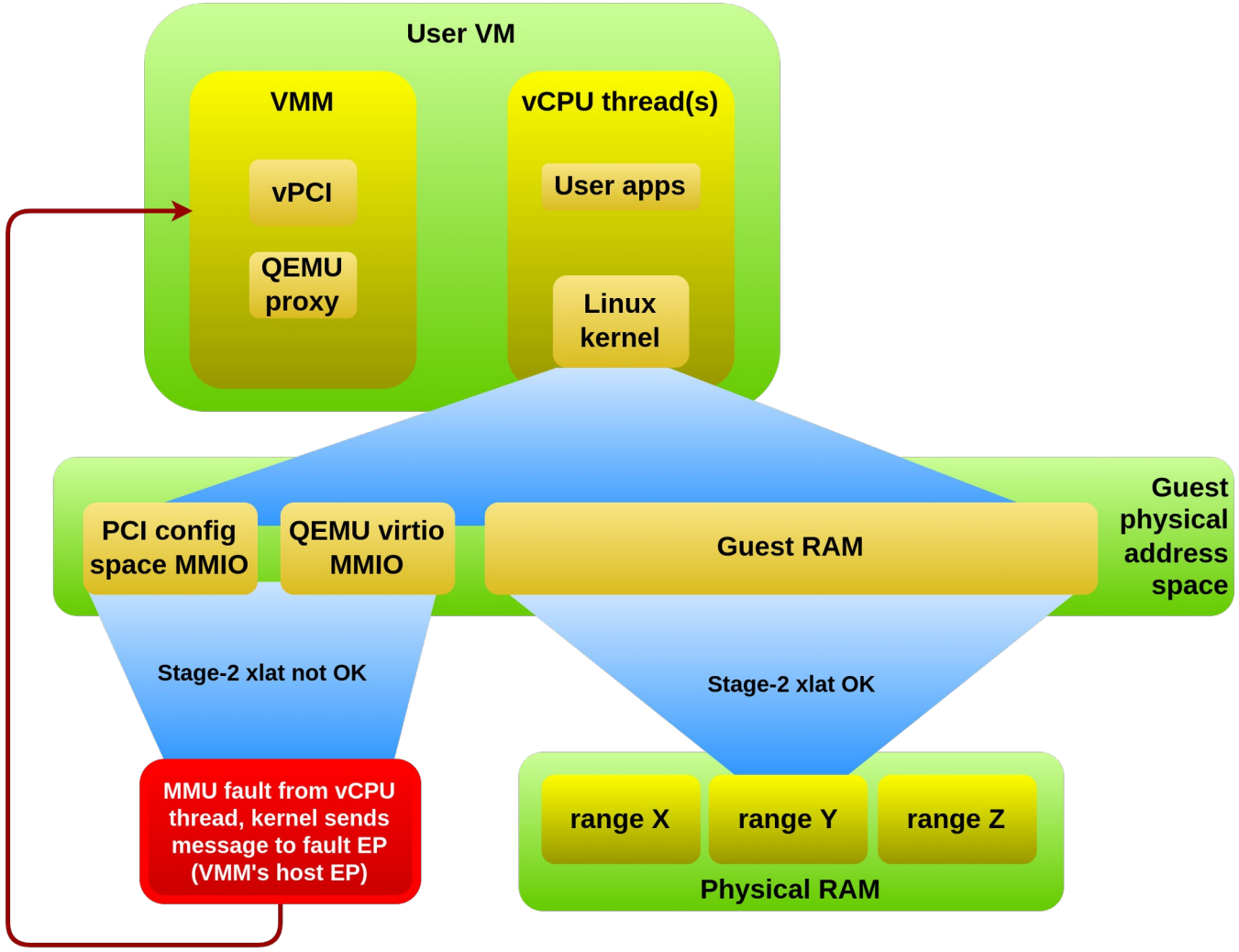
Secure Systems Research Centre

Virtio recap



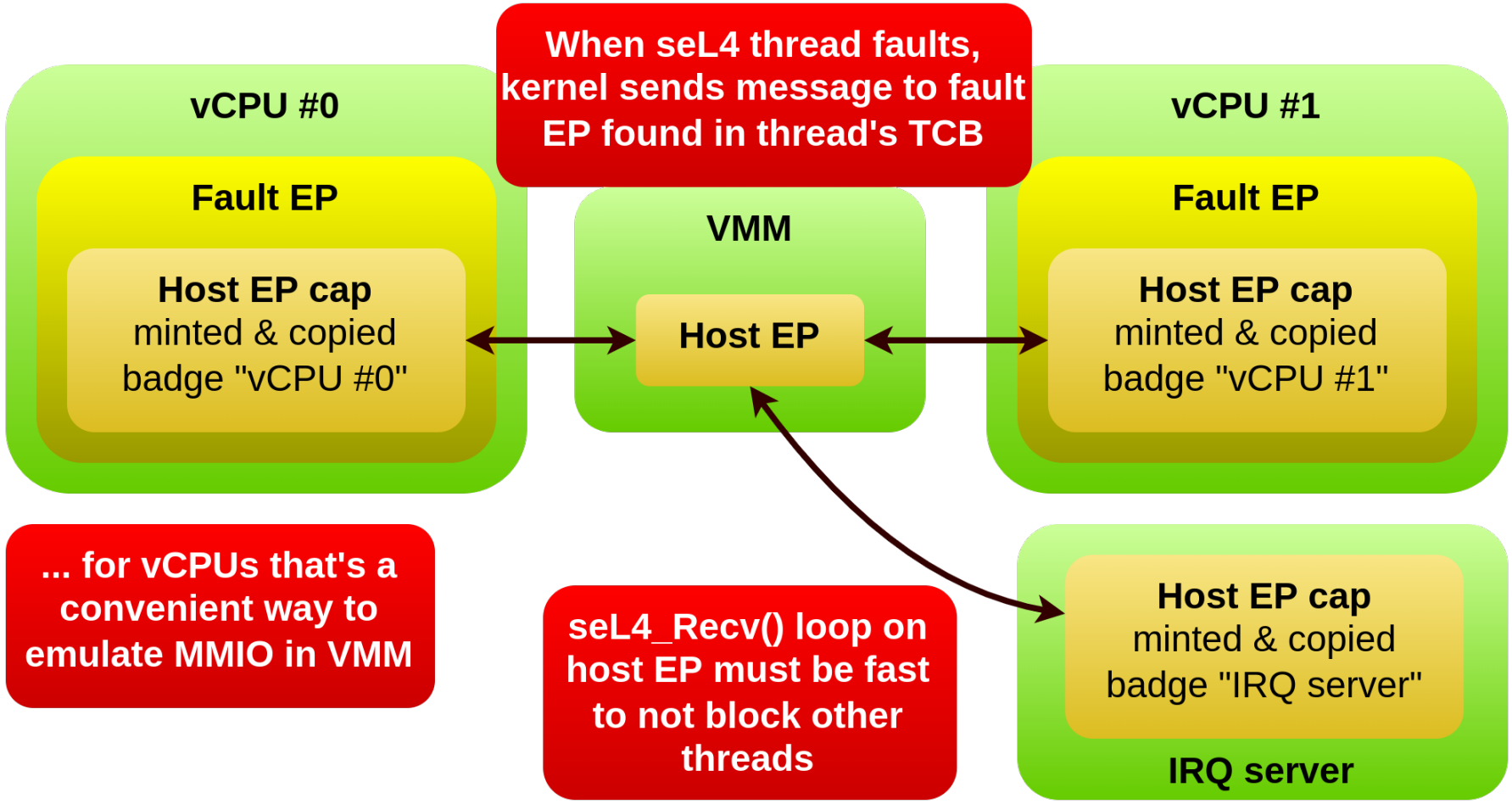
Secure Systems Research Centre

MMIO emulation



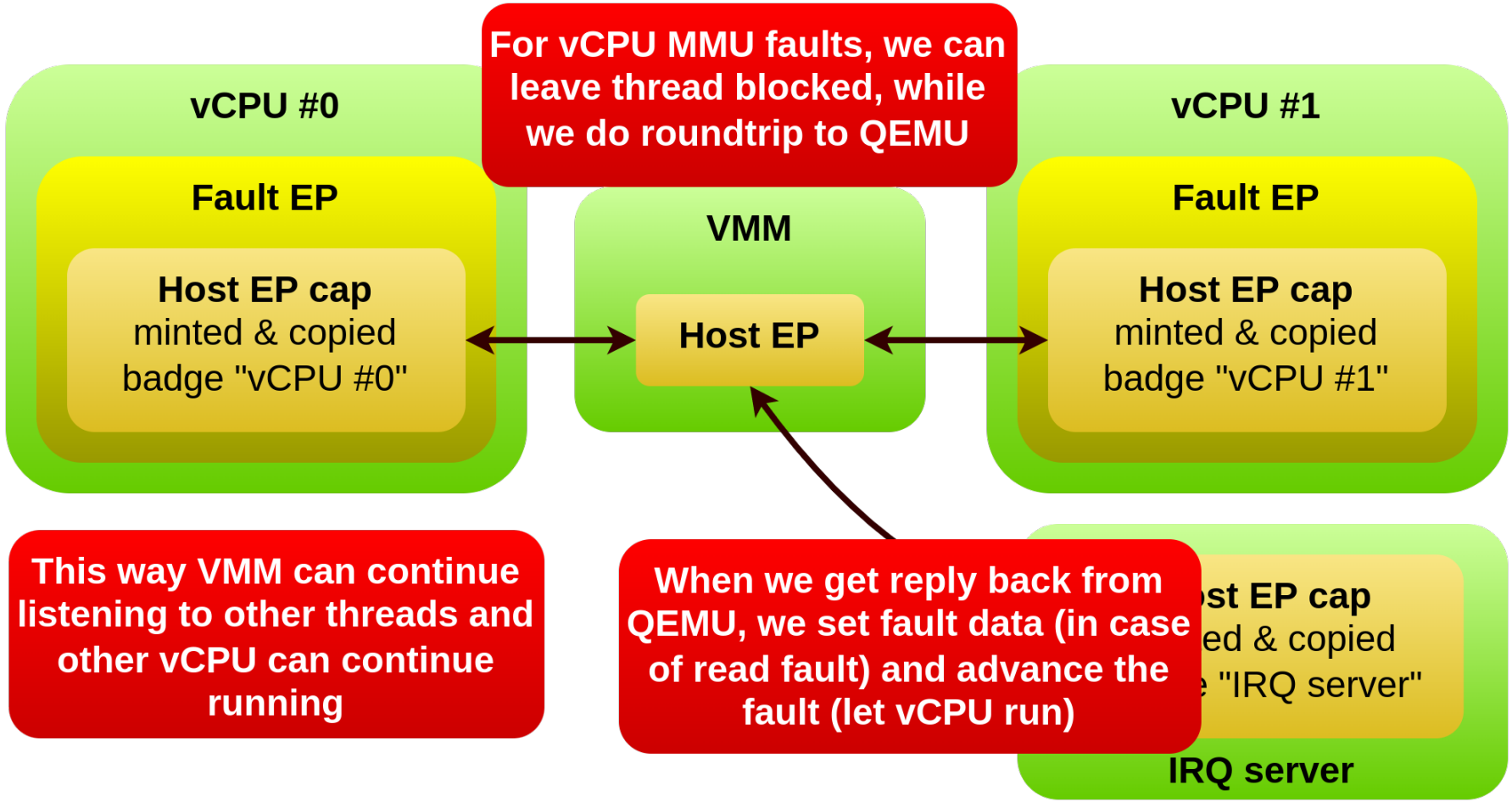
Secure Systems Research Centre

virtio MMIO



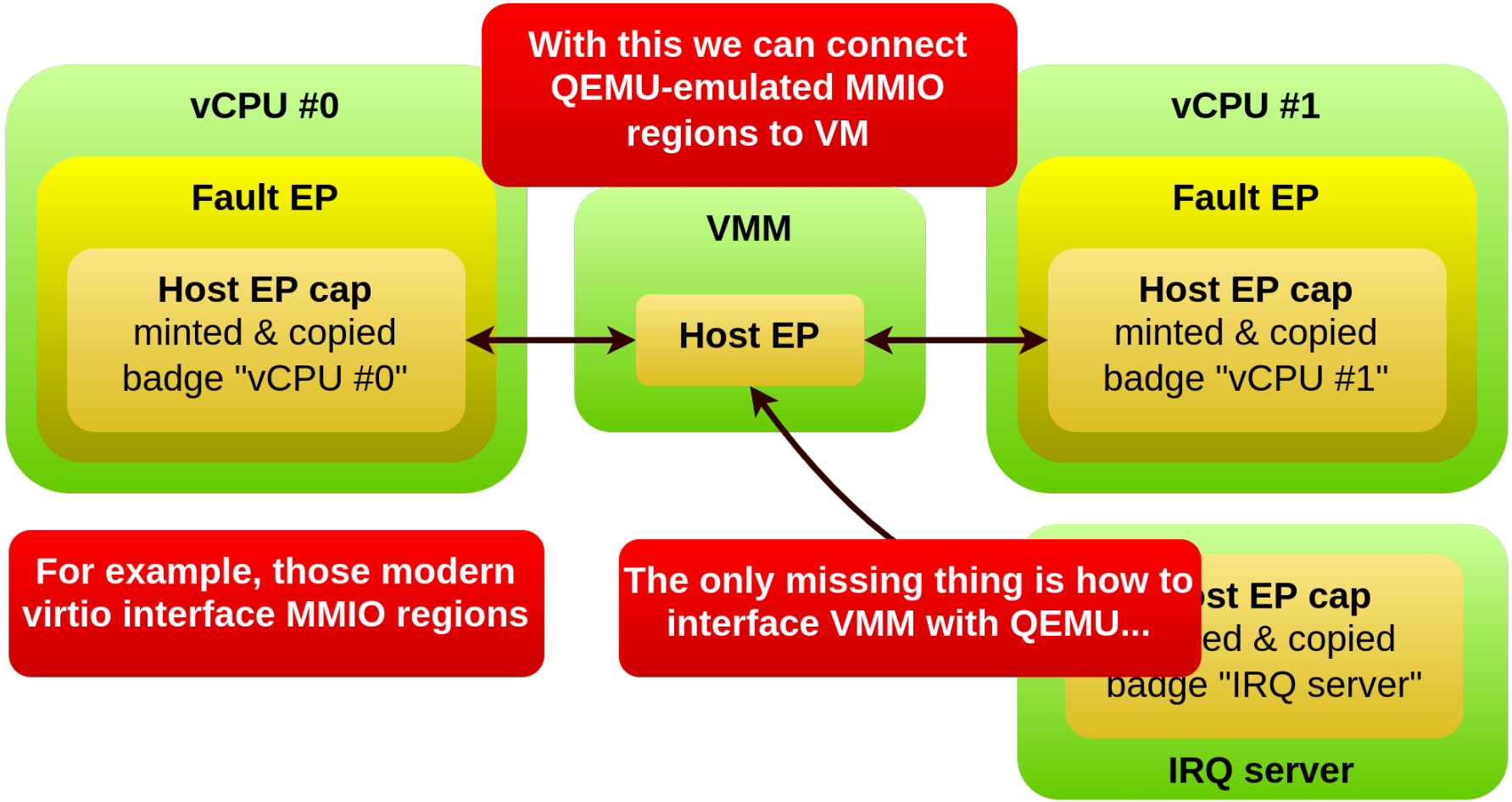
Secure Systems Research Centre

virtio MMIO



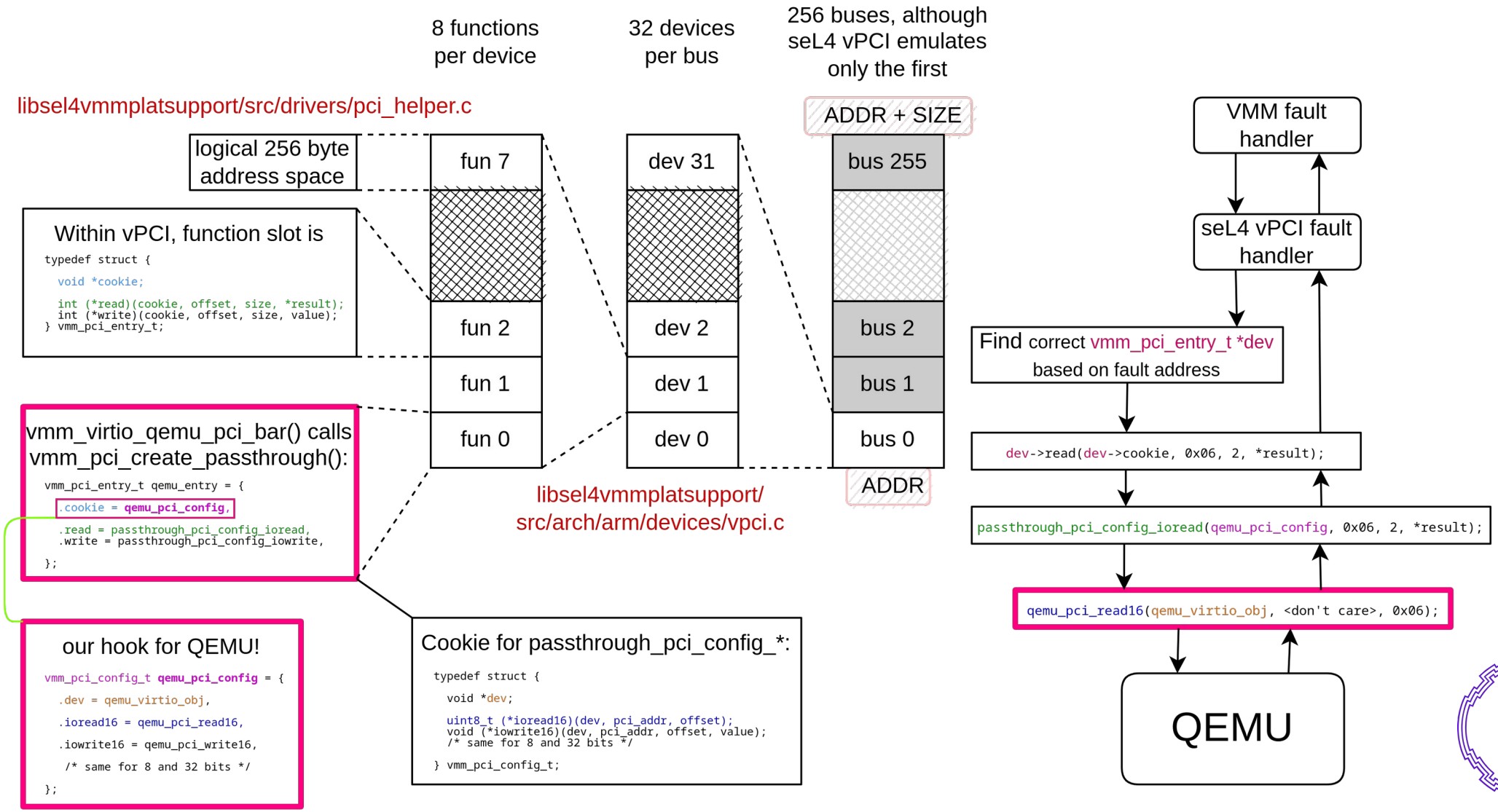
Secure Systems Research Centre

virtio MMIO



Secure Systems Research Centre

Virtual PCI in seL4



Secure Systems Research Centre

RPC

- To proxy faults to QEMU we need RPC
- Ring buffers on shared memory & doorbells
- On native side with seL4 threads everything is easy
- Shared memory is just (minted & copied) capabilities to frames
- Doorbells are just notifications

Secure Systems Research Centre

QEMU shared memory

- Static configuration
- VMM maps shared memory to VM's vspace (stage-2 mapping)
- VMM adds vPCI device whose BAR points to GPA of this mapping
- PCI driver kernel module is added to guest Linux
- Maps shared memory to userland via UIO (stage-1 mapping)
- We can map guest memory to QEMU's address space easily:

```
qemu-system-aarch64
  -object memory-back-end-file,\
    id=virt.ram,size=128M,mem-path=/dev/uio1,offset=4096,share=on\
  -machine memory-backend=virt.ram
```

- We had to add "offset" support
- Offset of 1 * PAGE_SIZE or 4096 selects dataport mapping instead of event BAR

QEMU doorbells (recv)

- Event BAR of cross-connector PCI device is the doorbell
- Mapped to QEMU via page 0 of /dev/uioX
- QEMU VMM thread blocks on UIO fd
- Host-VMM receives specially badged notification from guest-VMM
- Cross-connector increase event count in event BAR
- Injects IRQ to host VM
- Host’s guest Linux forwards IRQ to UIO
- UIO asks cross-connector whether this IRQ is for it
- Kernel module sets event count to zero (MMU fault)
- UIO unblocks QEMU thread doing poll() on fd

Secure Systems Research Centre

QEMU doorbells (send)

- Host Linux writes to event BAR emit register (MMU fault)
- Cross-connector sends seL4 notification
- Guest VMM notification wait thread unblocks
- Thread asynchronously (to VMM thread) handles ring buffer
- Async handler might post semaphore for ops than wait synchronously

Secure Systems Research Centre

Sharing VM RAM to QEMU

- Tweak init_ram_module() for user-VM
- Put shared memory frames where guest RAM frames used to be
- Those were unity mapping for ”untyped_mmios”
- We don’t really need unity mapping if we don’t use DMA
- Use shared memory frame caps to map memory to VM’s vspace

Secure Systems Research Centre

Level-triggered IRQs

- CAmkES VMM supports only edge-triggered IRQs
- QEMU’s virtio-pci needs level-triggered IRQs
- We added support for them
- Inject another IRQ if any external IRQ line still active

Secure Systems Research Centre

WFI / seL4_Yield()

- Important to get thread priorities right
- ”Host” threads should have higher priority than ”guest”
- VMM threads should have higher priority than vCPU threads
- Host continues to execute after QEMU has served the guest
- Host just executes WFI (Wait For Interrupt) until its timeslice is consumed
- Our modification to seL4 kernel does yield when it traps WFI from user threads
- virtio-blk benchmark shows 25 % – 40 % improvement in throughput

Secure Systems Research Centre

seL4/guest schedulers

- Two levels of schedulers
- One in seL4 kernel (for each CPU)
- Guests running their own schedulers
- Nasty interactions
- Big field of research

Secure Systems Research Centre

Tracing

- Essential for improving performance
- sel4bench good for kernel and native threads
- We need also traces from guest VMs (Linux kernel / QEMU)
- seL4 kernel / VMM tracing easy
- Guest tracing is more complex since there can be multiple vCPUs
- Hard to implement atomic writes to trace buffer
- Let’s avoid “not invented here” and use Linux ftrace instead

Tracing

- We intend to convert seL4 kernel and VMM traces into ftrace format
- Merging kernel, VMM and guest traces into one
- Possibly existing work from KVM community can be used
- We need to record virtual timer offsets on vCPU thread switches to do that
- After that we can use kernelshark, FlameGraphs, etc.
- The whole system will be visualized
- ”Doing nothing” or ”doing context switches” more easily pinpointed

ioeventfd

- Every QEMU hypervisor backend has to implement memory listeners
- Configures callbacks that get called when given region is accessed
- Our prototype uses custom code
- Similar thing exists in KVM, called ioeventfd
- ioeventfds configured on /dev/kvm with proper ioctl()
- Better change our code to adhere to this API
- Prepares us for other KVM-compatible VMMs
- Hides the memory listener details from VMM

Secure Systems Research Centre

irqfd

- QEMU needs a way to raise IRQ in guest
- Once again, our code is one-off hack
- Glues the injection code to QEMU’s gpex PCI host
- KVM has irqfd concept
- When you write to fd, IRQ gets injected to guest
- Supporting irqfds gives us even more KVM compatibility

Secure Systems Research Centre

Dynamic RAM allocation

- With pure virtio VMs we don’t need DMA capability
- No need for unity stage 2 mapping
- We can allocate RAM from VM anywhere
- GPA range stays the same due to stage 2 mapping
- Idea: create a ”mallocator” seL4 application
- Has a pool of untyped memory
- VMMs request memory from mallocator
- It copies/mints capabilities and transfers them to VMM
- VMM maps these caps as VM’s RAM

Secure Systems Research Centre

Dynamic VMs

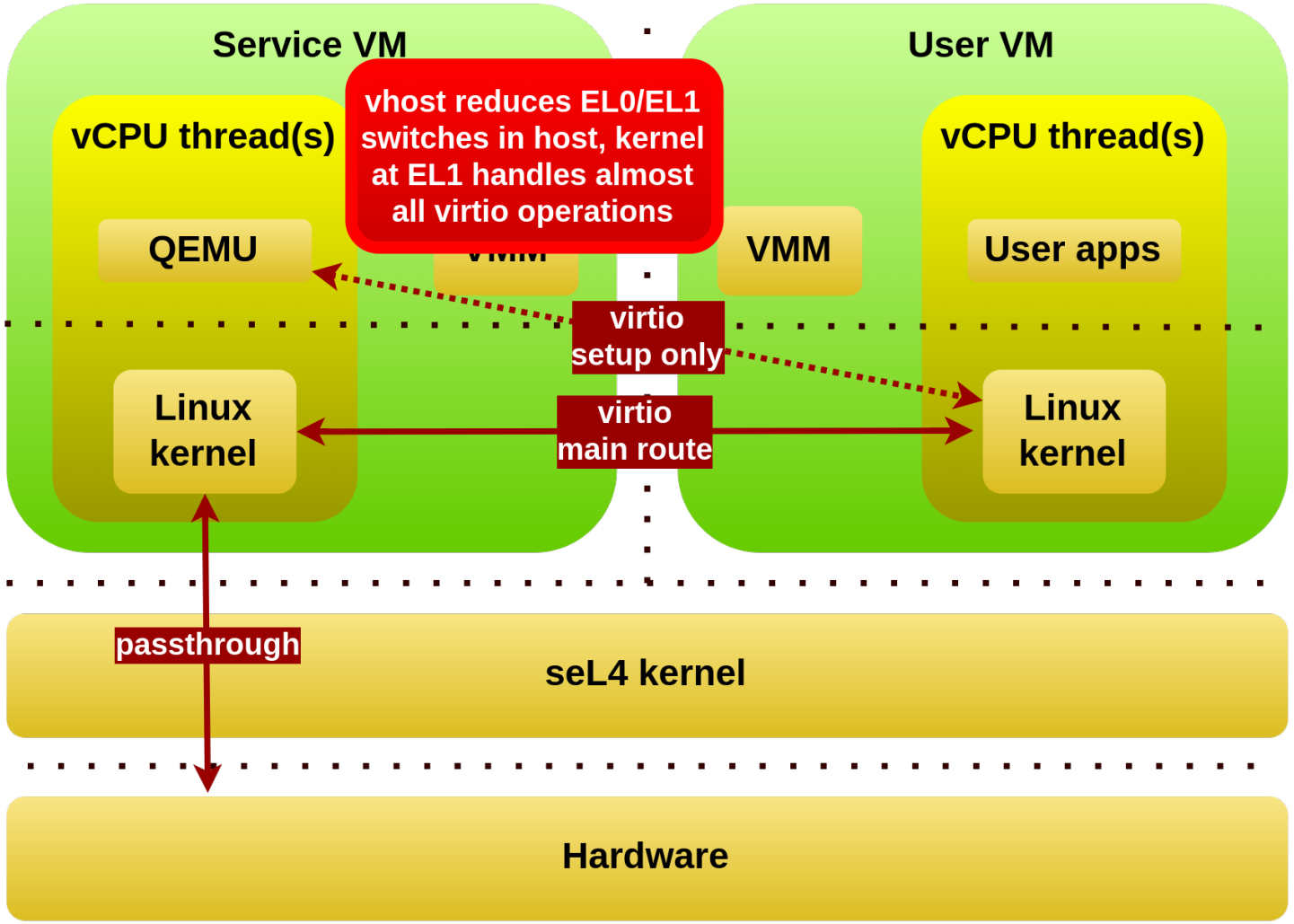
- ioeventfd and irqfd simplifies VMM porting
- Dynamic RAM allocation doable
- Some more bits from KVM API need to be provided
- We need to provide means to create vCPUs, route IRQs etc.
- Most probably we won’t be able to be 100% pin-compatible…
- … but not sure if that’s even desirable
- The important point is to minimize the VMM porting effort
- CrosVM, firecracker, etc.

Secure Systems Research Centre

Host blinding

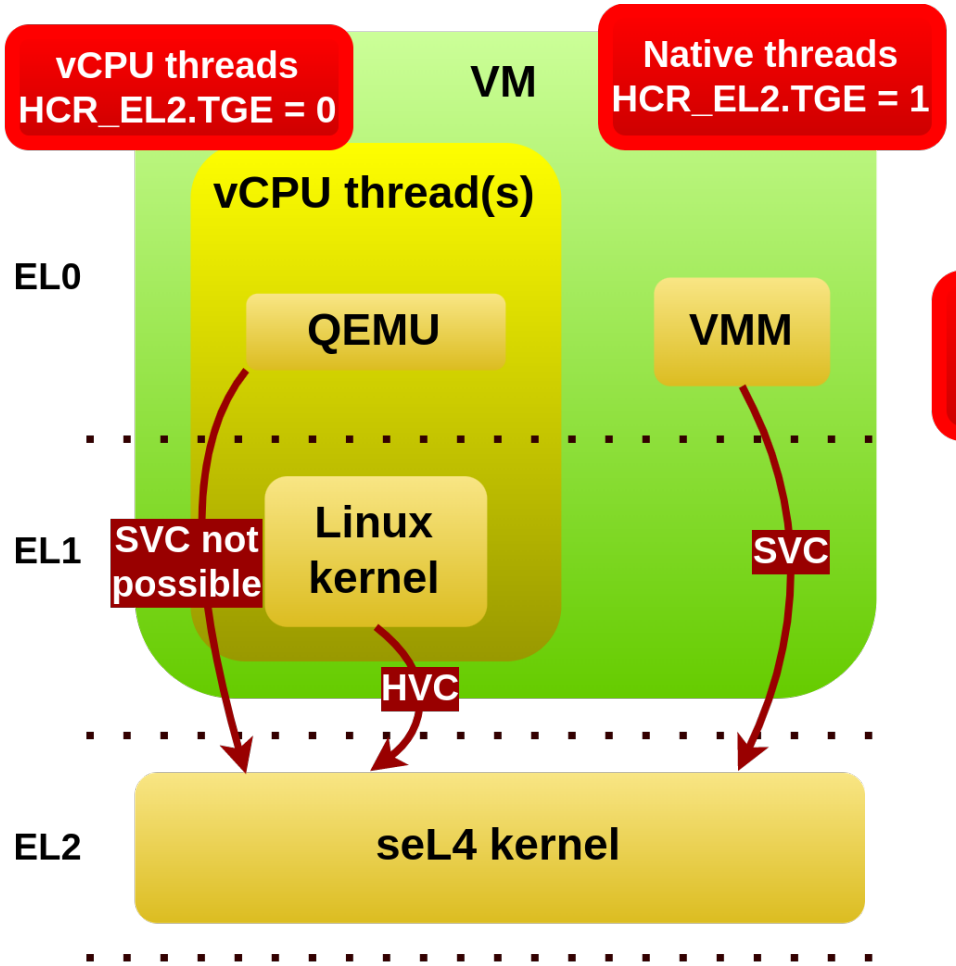
- Right now host sees all of the guest's RAM
- Only pages involved with virtio needed
- Idea: use mallocator to restrict host access
- Mallocator intercepts notifications between host and guest
- Tracks which pages guest wants to share to host
- Host VMM can ask mallocator to give caps to virtio pages

vhost



Secure Systems Research Centre

seL4 virtio transport

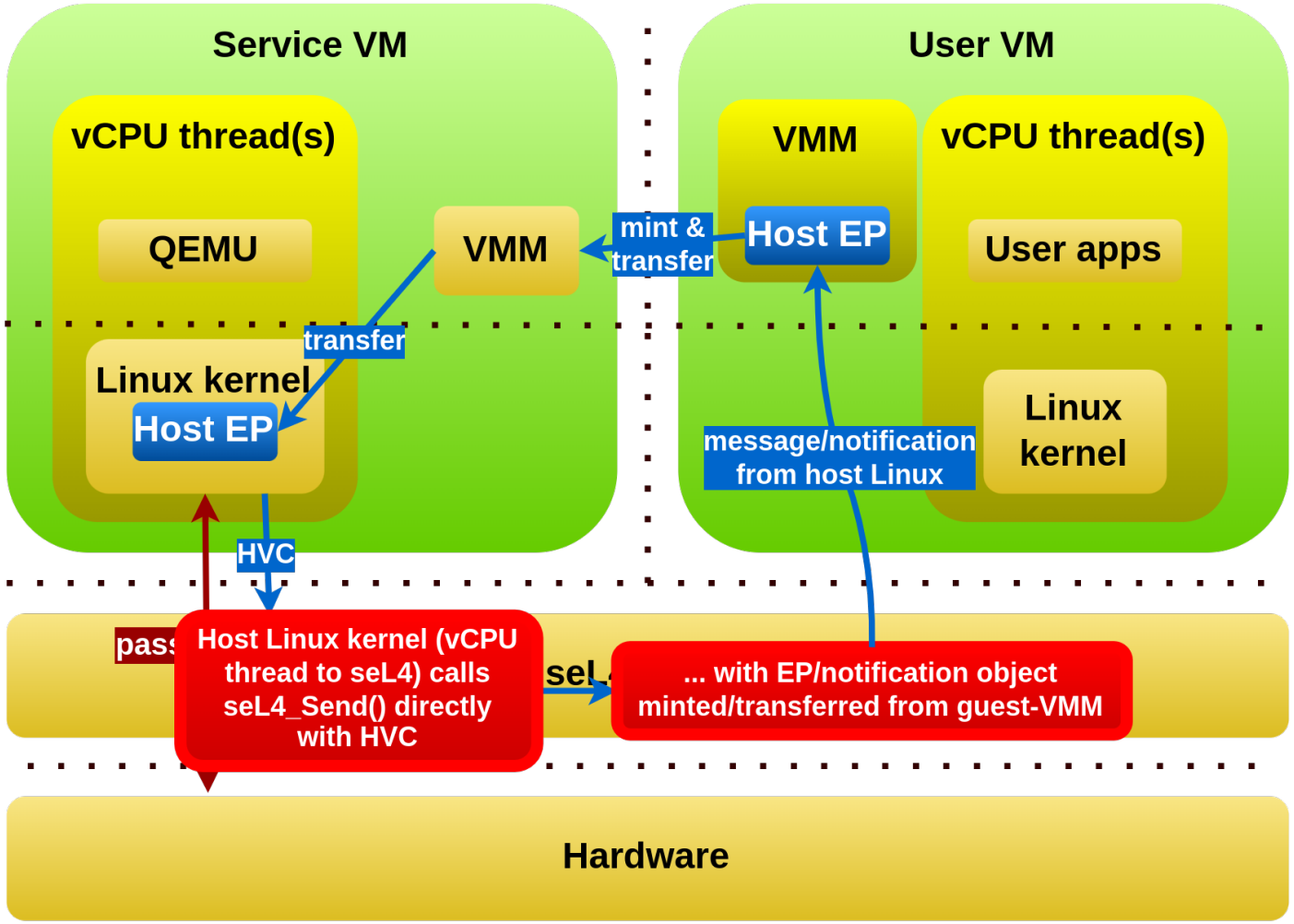


QEMU can call seL4_Send() only
a) via Linux kernel
b) via VMM-handled MMU fault

Either way, needless
context switch

Secure
Systems
Research
Centre

seL4 virtio transport



Secure Systems Research Centre


Technology
Innovation
Institute