TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Betriebssysteme

THE NOVA KERNEL API

Julian Stecklina (jsteckli@os.inf.tu-dresden.de)

Dresden, 5.2.2012

00 Disclaimer

This is not about OpenStack Compute.

NOVA is mainly the work of Udo Steinberg (kernel) and Bernhard Kauer (userland).

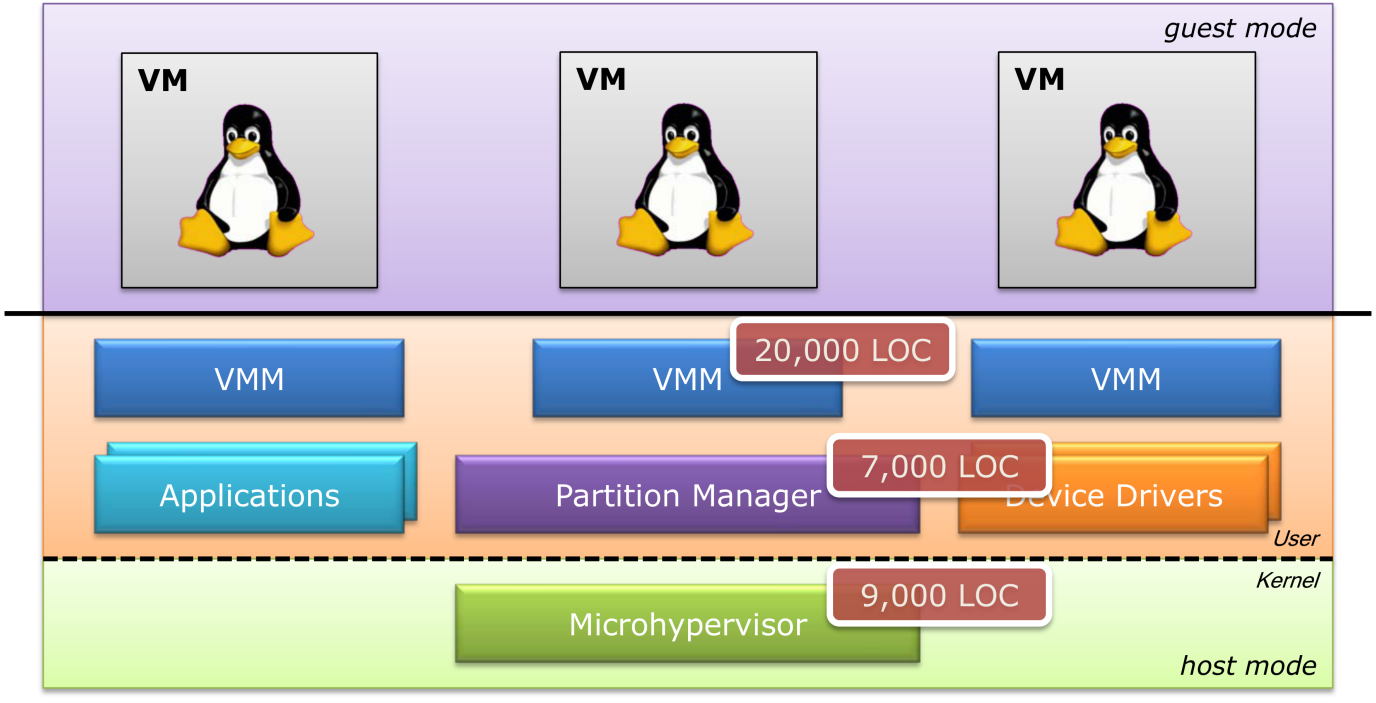
`http://hypervisor.org/`

00 Goals

- not talking about virtualization propaganda,
- giving a very short overview of NOVA as a whole
- introducing basic concepts of the kernel API

In the end you should be able to pick up the NOVA API manual and make heads or tails of it.

01 NOVA OS Virtualization Architecture



http://os.inf.tu-dresden.de/papers_ps/steinberg_eurosys2010.pdf

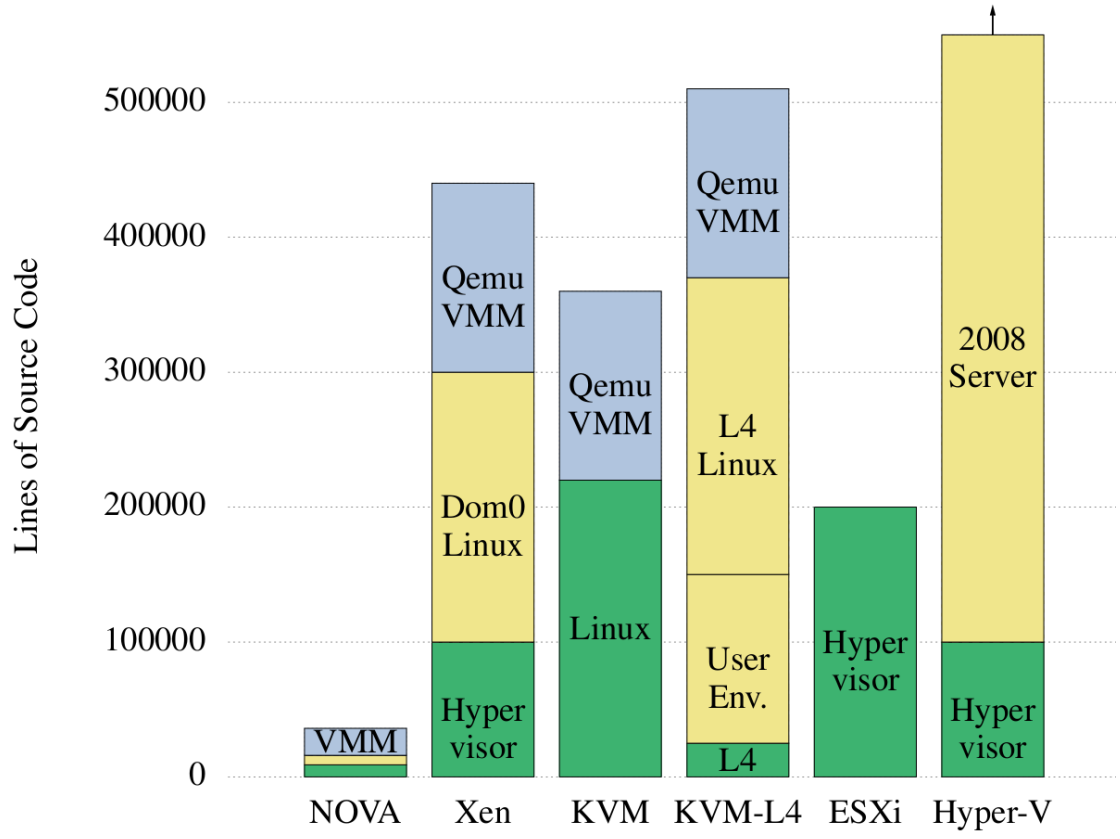
01 What works, what doesn't

Works

- x86 32-bit
- SMP
- VT-x, AMD-V
- VT-d (Intel IOMMU)
- SR-IOV
- grub, syslinux, . . .
- Linux, L4, . . .
- emulates AHCI, igb, . . .
- drivers for AHCI, some Intel NICs, . . .
- experimental libvirt support

Doesn't work yet

- Windows
- Migration
- Recursive Virtualization
- 64-bit
- being user-friendly ;-)
- . . .


Lines of Source Code
500000
400000
300000
200000
100000
0
VMM
Hyper visor
Qemu VMM
Dom0 Linux
Qemu VMM
Linux
Qemu VMM
L4 Linux
User Env.
L4
Hyper visor
2008 Server
Hyper visor
NOVA
Xen
KVM
KVM-L4
ESXi
Hyper-V

02 NOVA Architecture

Reduce complexity of hypervisor:

- hypervisor provides low-level protection domains
  - address spaces
  - virtual machines
- one VMM per guest in (root mode) userspace,
  - possibly specialized VMMs to reduce attack surface
  - only one generic VMM implement so far

Demo

03 The L4 Influence

NOVA cannot deny its roots in the L4 family:

- task, threads, synchronous IPC
- recursive mapping of memory

03 Capability-Based

Syscalls operate on capabilities to kernel objects:

- Protection Domain (PD) ("task") — `create_pd`
- Execution Context (EC) ("thread") — `create_ec`, `ec_ctrl`
- Scheduling Context (SC) — `create_sc`, `sc_ctrl`
- Portals (PT) — `create_pt`, `call`, `reply`
- Semaphore (SM) — `create_sm`, `sm_ctrl`

03 Capabilities

Userspace can

- create capabilities to objects (by creating kernel objects),
- delegate capabilities (recursively, just as memory),

Capabilities are stored per-PD in capability space in the kernel. A PD

- uses index into capability space to name capabilities,
- unforgeable.

(Think file descriptors.)

03 Communication

EC (thread)

- bound to one PD (address space)
- either thread or vCPU
- has a special memory region (UTCB) for IPC

Portals

- entry point (instruction pointer)
- bound to one EC
- per client/function/...
- pass data, delegate capabilities from UTCB to UTCB
- can be *call*ed or implicitly used by exceptions (if a thread has the cap)

03 ECs and SCs

There are two kinds of threads:

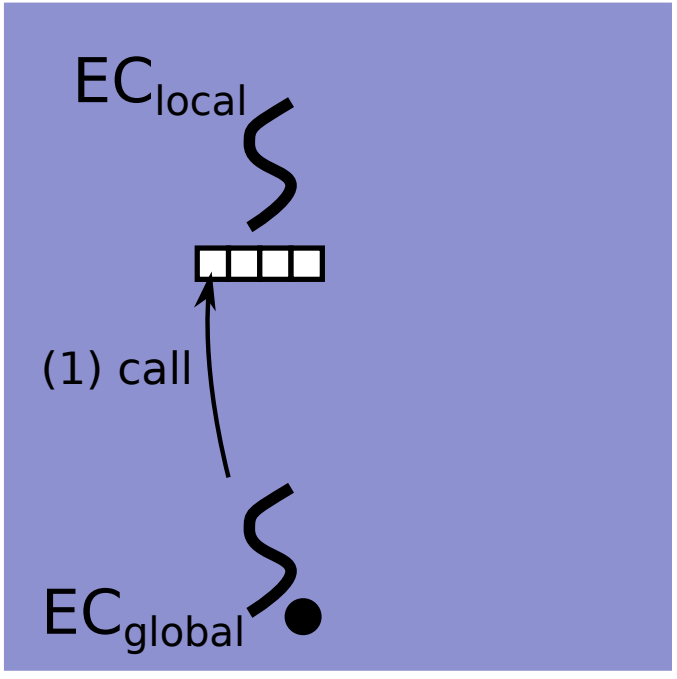
with time

- "global thread" or vCPU
- stick SC to newly created EC
- causes startup exception when first scheduled

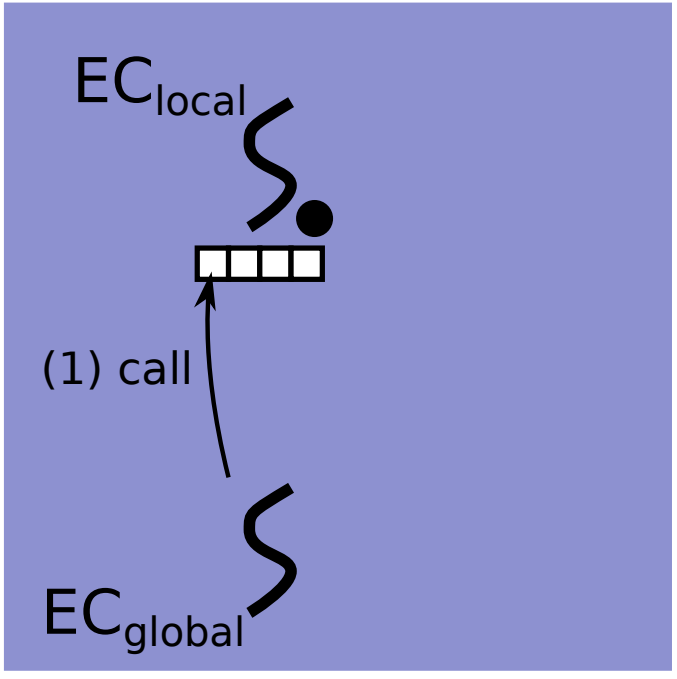
without time

- "local thread"
- bind portals to ECs
- when portal invoked, starts executing at portal EIP
- caller hands in time to handle the request (no scheduling decision)

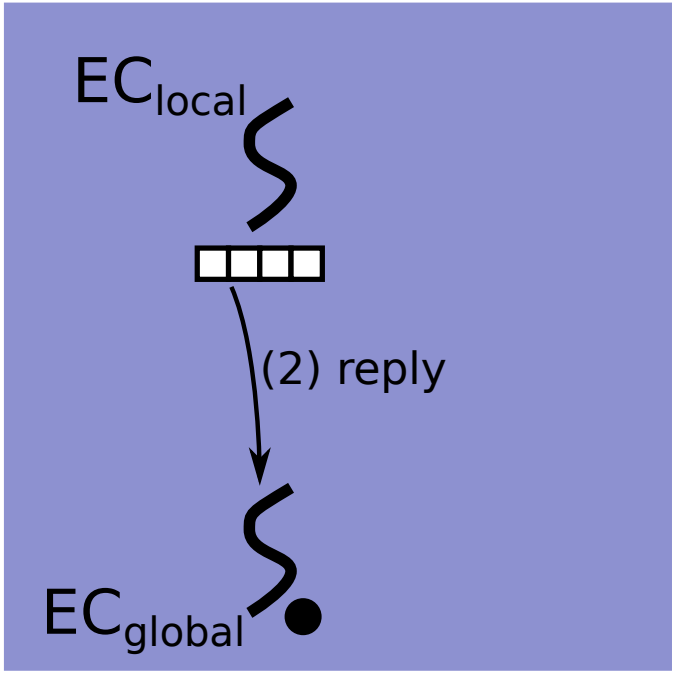
03 Basic Server Scenario



03 Basic Server Scenario



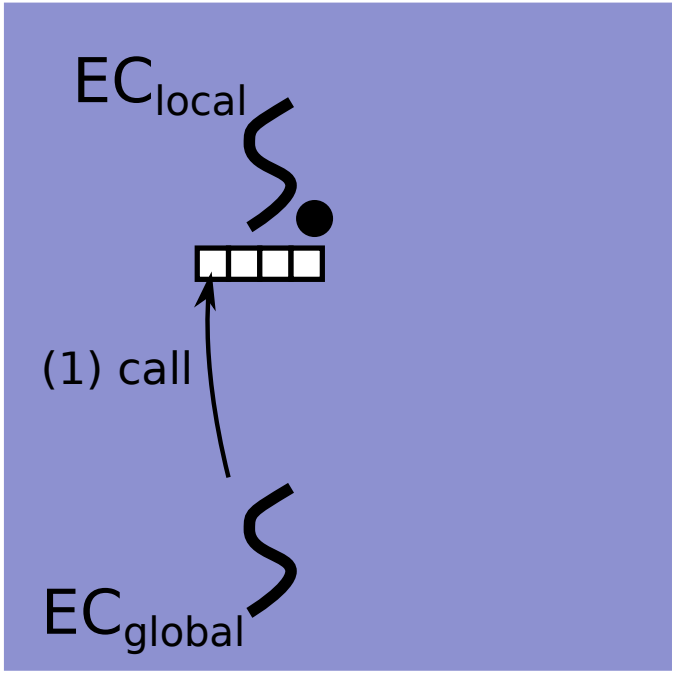
03 Basic Server Scenario



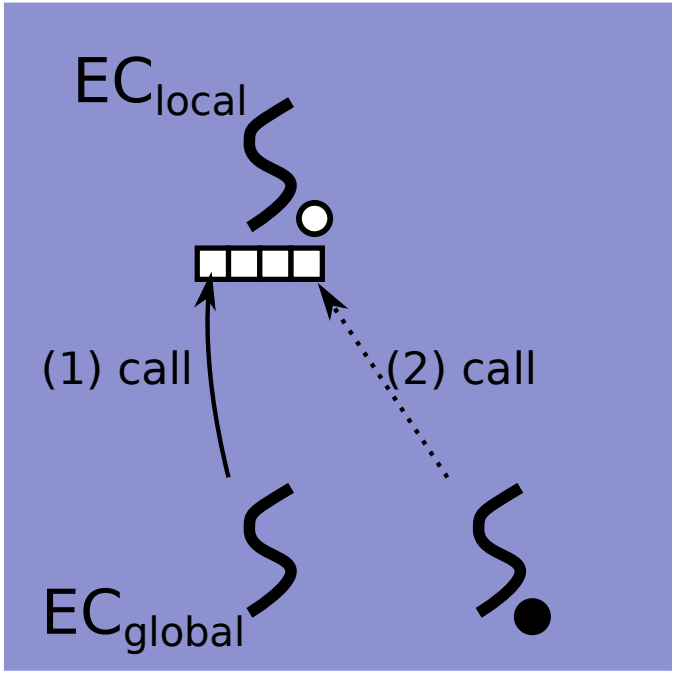
03 Resource Contention

ECs are not reentrant.
What happens when a second client wants to call a service?

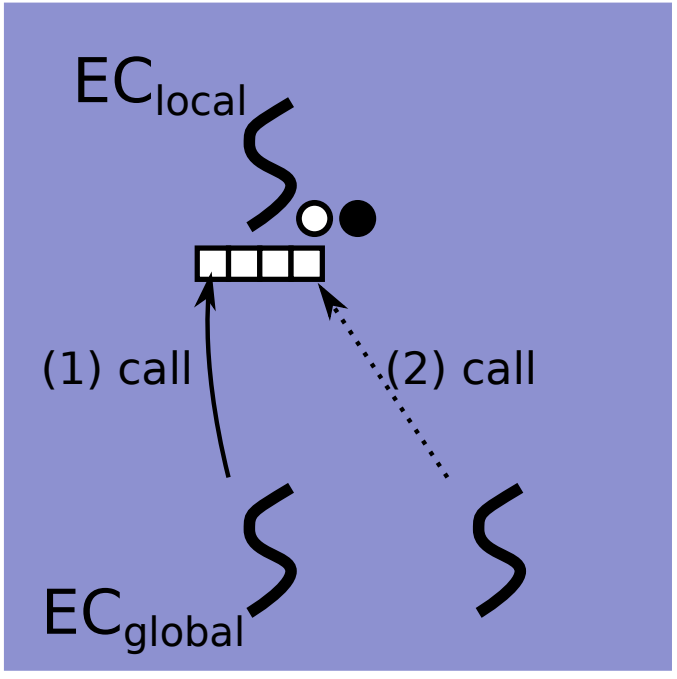
03 Resource Contention



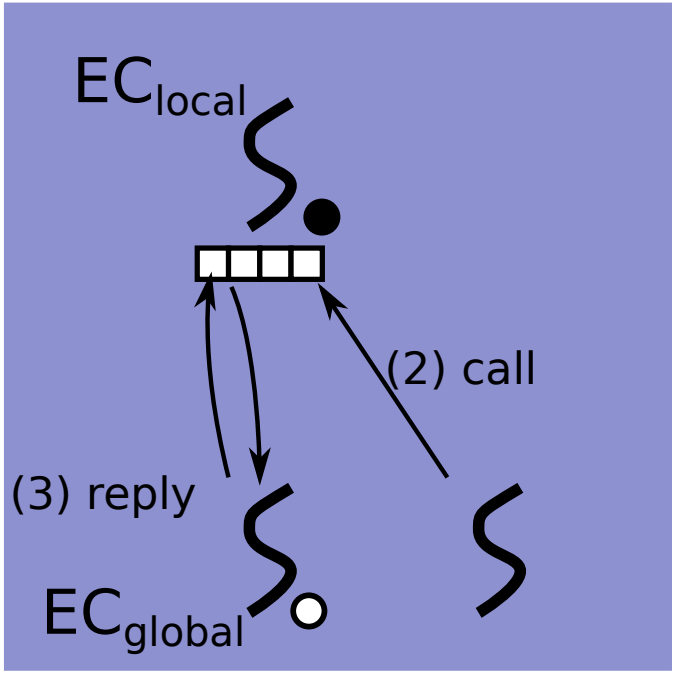
03 Resource Contention



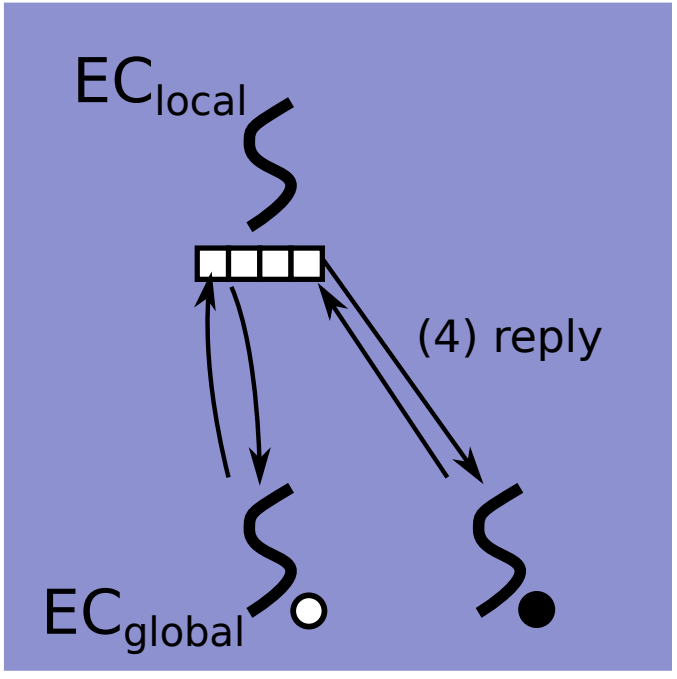
03 Resource Contention



03 Resource Contention



03 Resource Contention



03 NOVA's time management

With SCs only bound to some threads, it is possible to build (simple) servers without time reservation.

- How much time should service *foo* need anyway?
- fewer things to schedule,
- contended resources get "boosted" by clients as needed.

04 Hardware Support for Virtualization

Late Pentium 4 (2004) introduced hardware support for virtualization: Intel VT.
(AMD-V is conceptually very similar)

- root mode vs. non-root mode
  - root mode runs hypervisor
  - non-root mode runs guest
- situations that Intel VT cannot handle trap to root mode (**VM Exit**)
- special memory region (VMCS) holds guest state
- reduced software complexity

Supported by all major virtualization solutions today.

04 VT-x Problems

VMCS (memory region holding guest state) needs to manipulated by VMM, yet

- cannot be mapped into userspace,
- have to use privileged VMREAD/VMWRITE instructions to access,
- reading all content for every VM Exit is expensive.

Kernel has to manage VMCS access.

04 Virtualization on NOVA

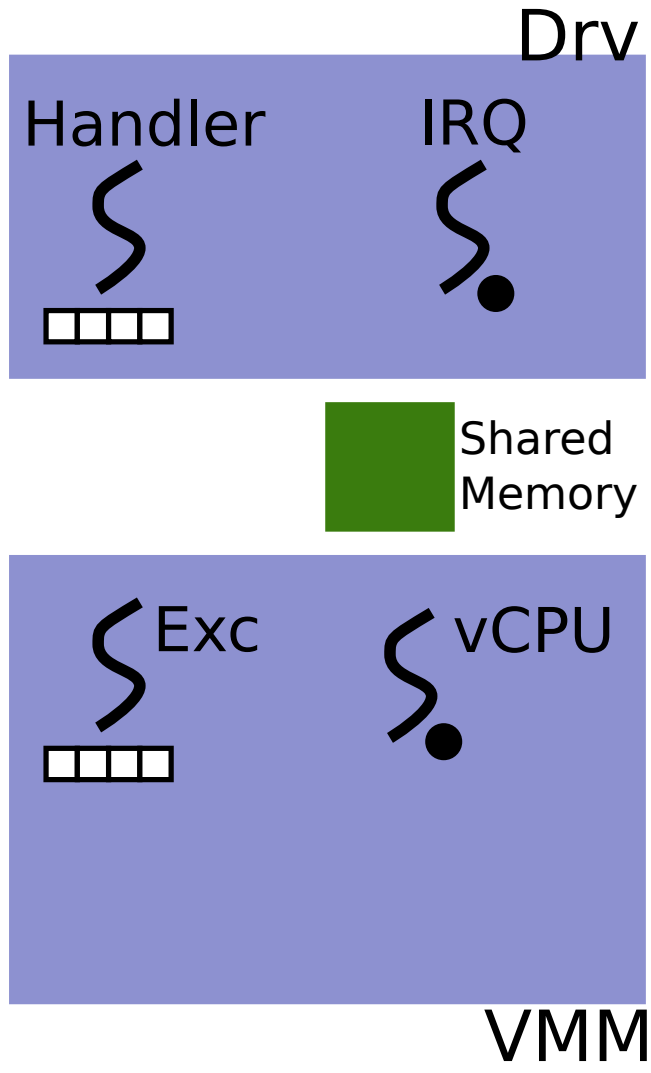
VM Exits (and normal exceptions) vector through special portals.

- Portals created with bit field denoting interesting information (Message Transfer Descriptor, MTD)
  - for WRMSR or CPUID we need only general purpose registers
  - for page fault we need complete vCPU state
- kernel puts this data in handler's UTCB
- handler produces new MTD on reply

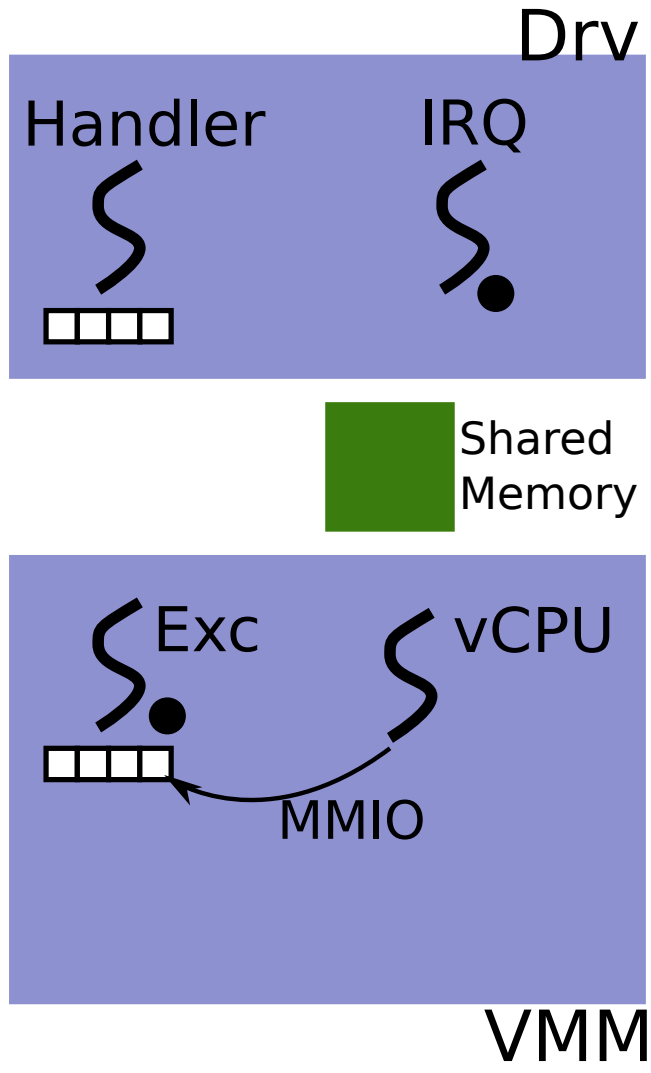
Reduce number of expensive VMREAD/VMWRITE in the kernel.

04 Writing to disk

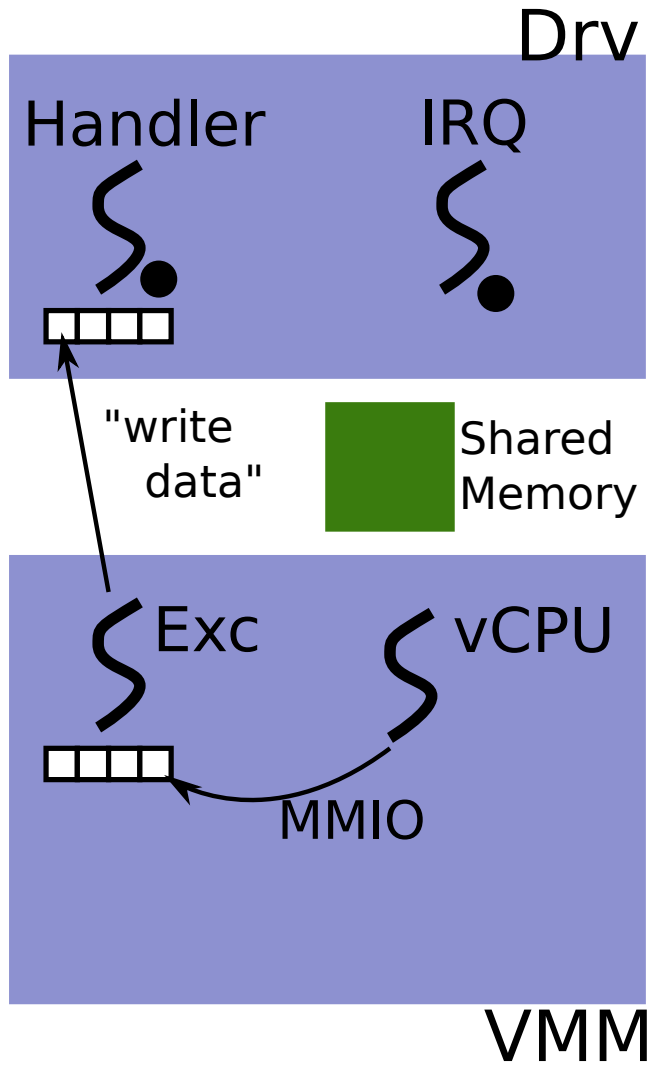
04 Writing to disk



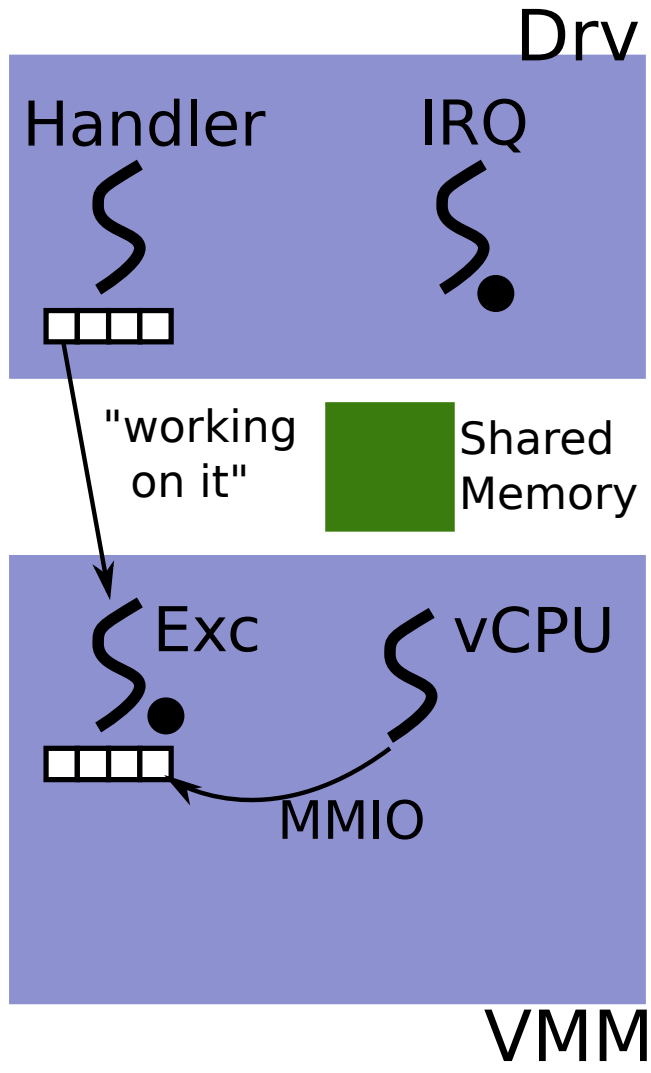
04 Writing to disk



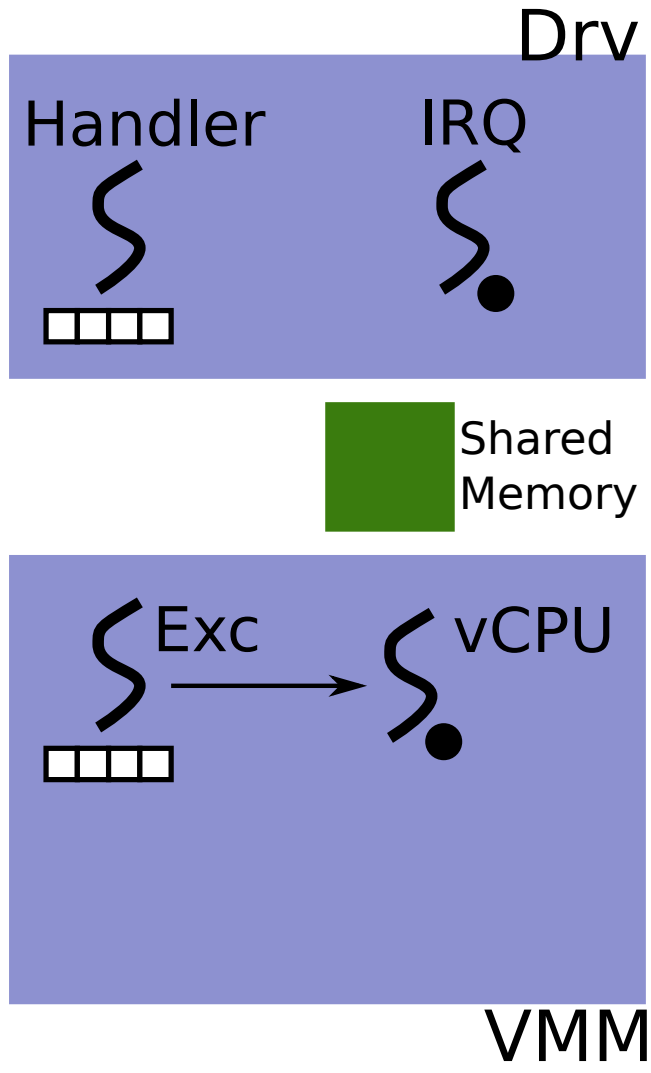
04 Writing to disk



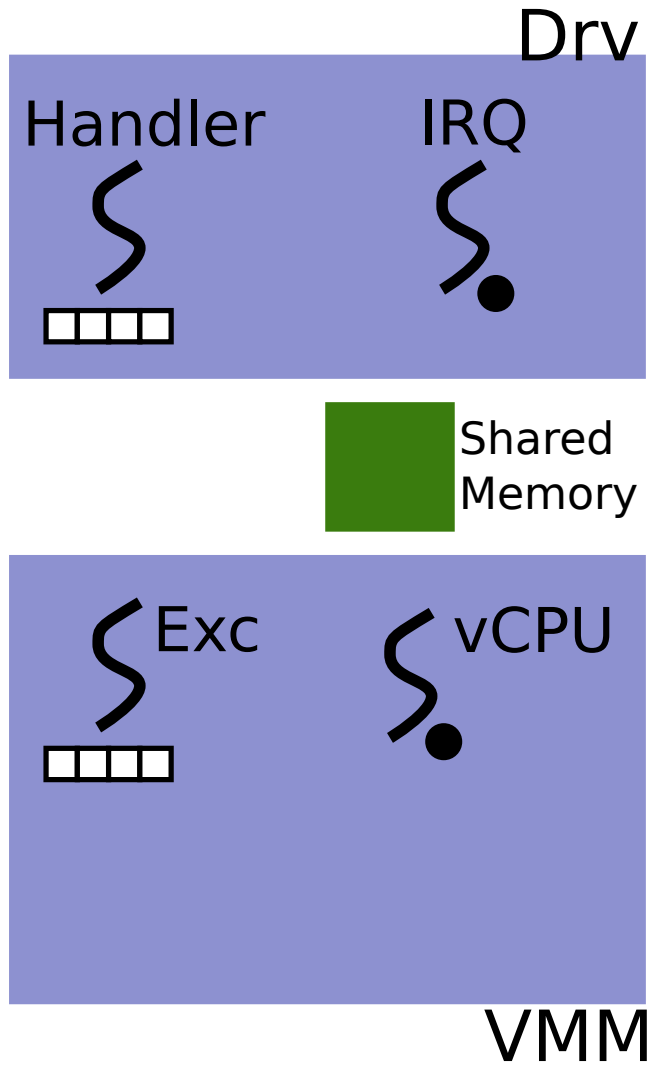
04 Writing to disk



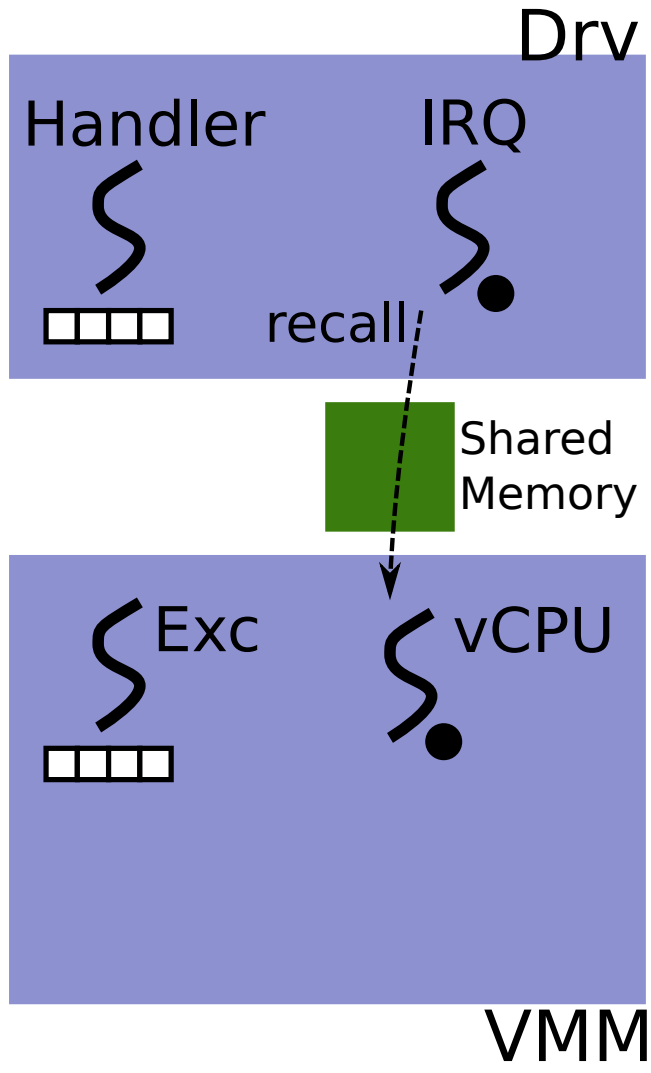
04 Writing to disk



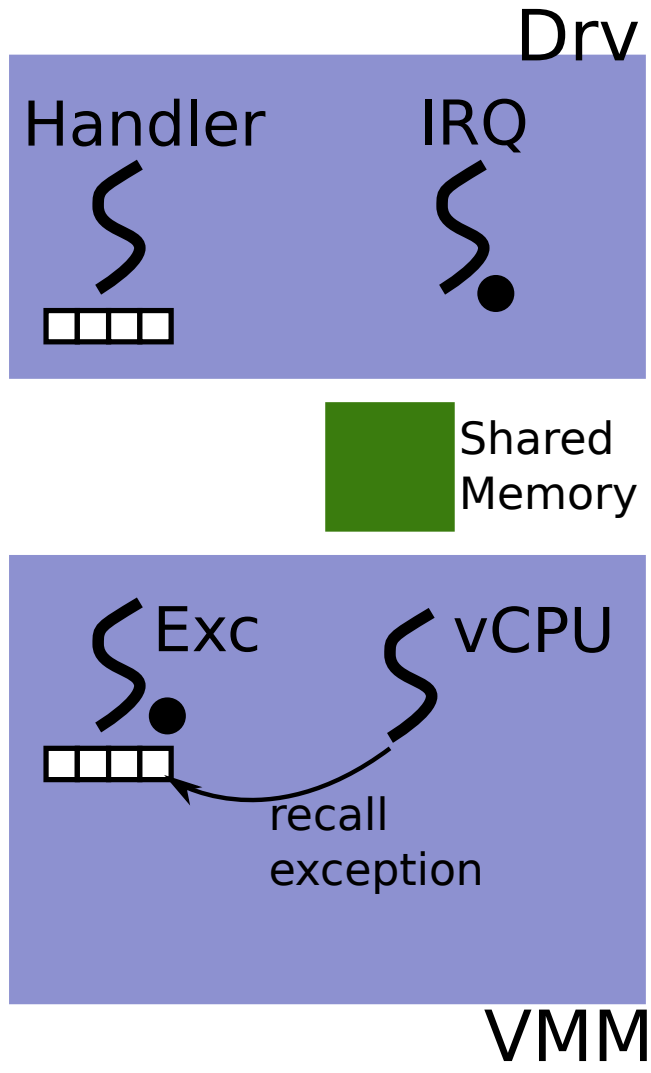
04 Writing to disk



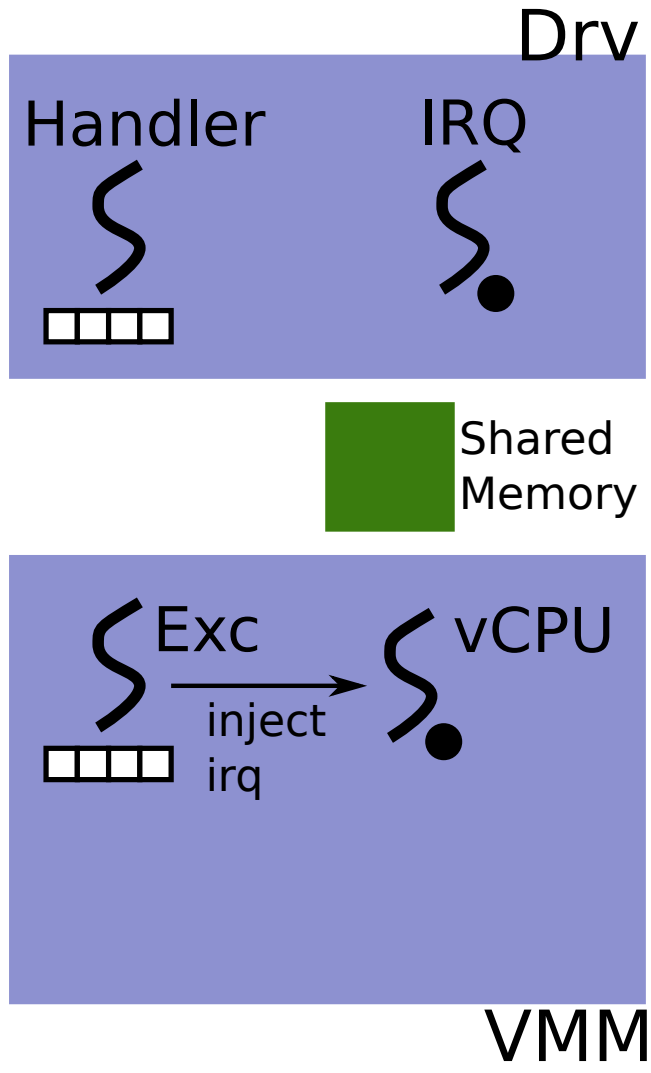
04 Writing to disk



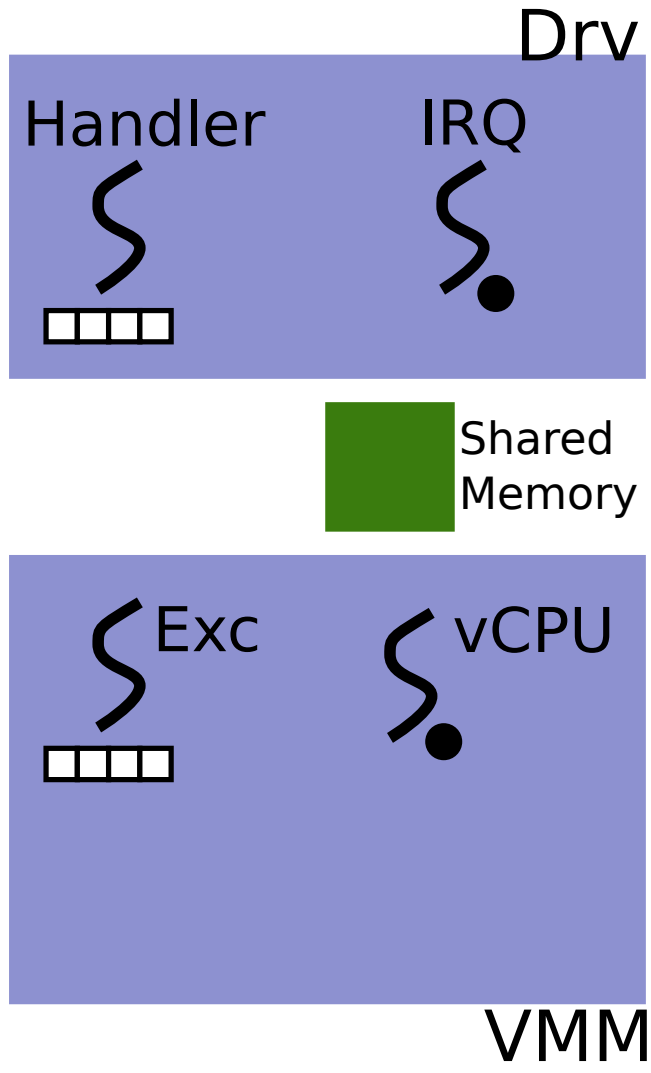
04 Writing to disk

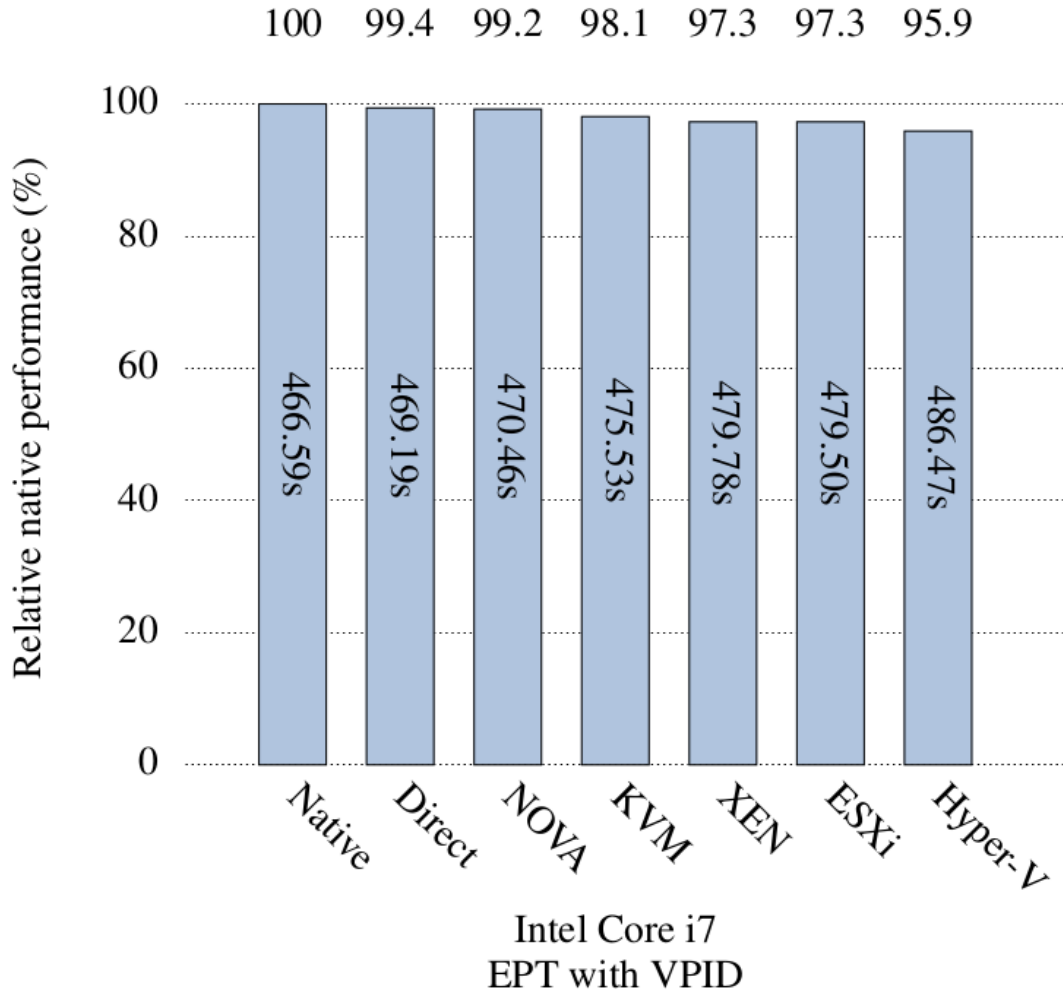


04 Writing to disk



04 Writing to disk





http://os.inf.tu-dresden.de/papers_ps/steinberg_eurosys2010.pdf

05 There is also . . .

- Userspace Timer Service
- Admission Server
- Device Drivers (IOMMU!)
- . . .

05 Summary

The NOVA microhypervisor is a

- fast capability-based microkernel
- with virtualization

in mind.

Supported by:

http://ict-passive.eu/

Code at http://hypervisor.org/

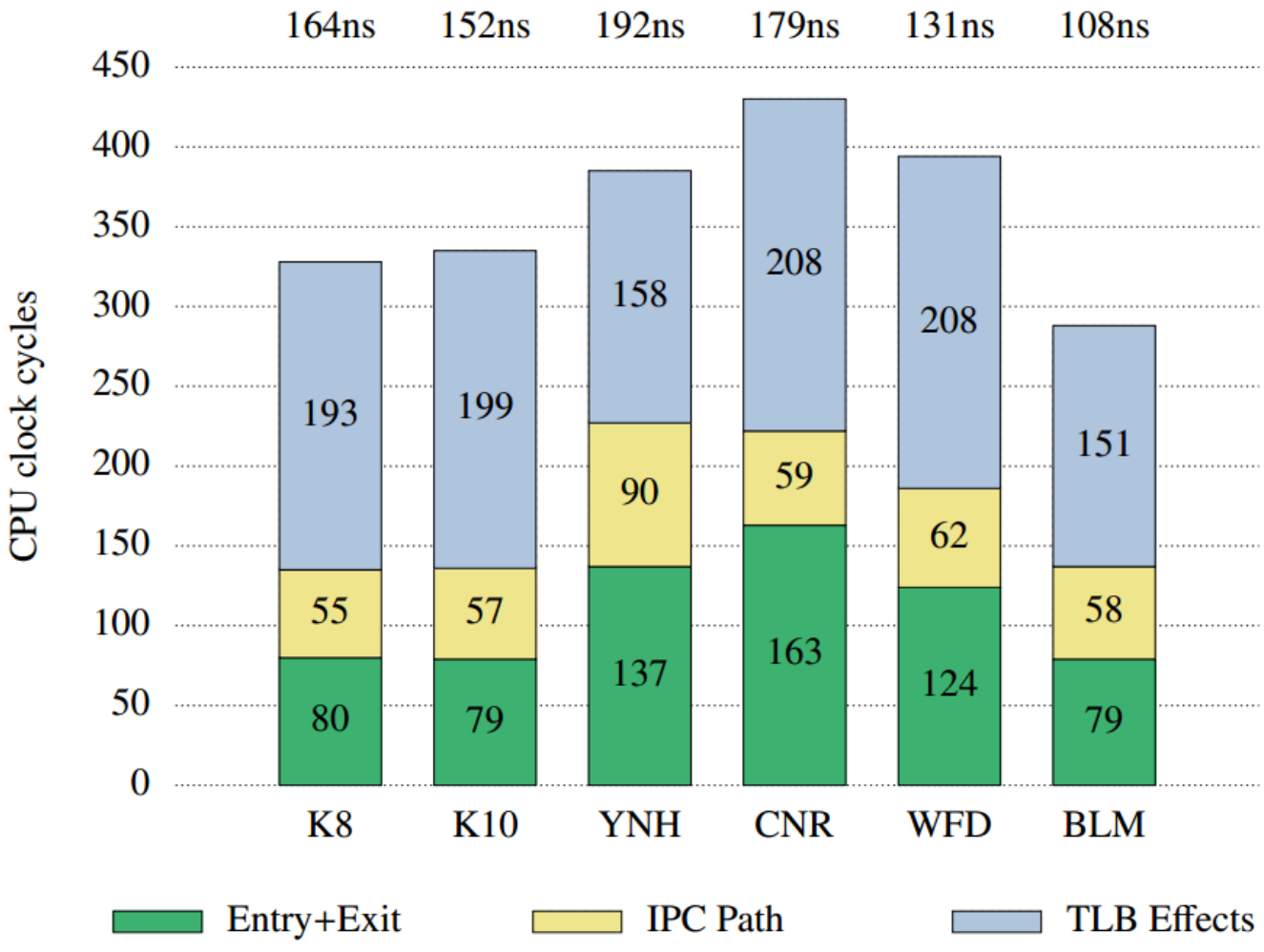
Discuss at http://os.inf.tu-dresden.de/mailman/listinfo/l4-hackers

06 Multiple CPUs

Thread-related kernel objects are bound to one CPU:

- Portals,
- Execution Contexts,
- Scheduling Contexts.

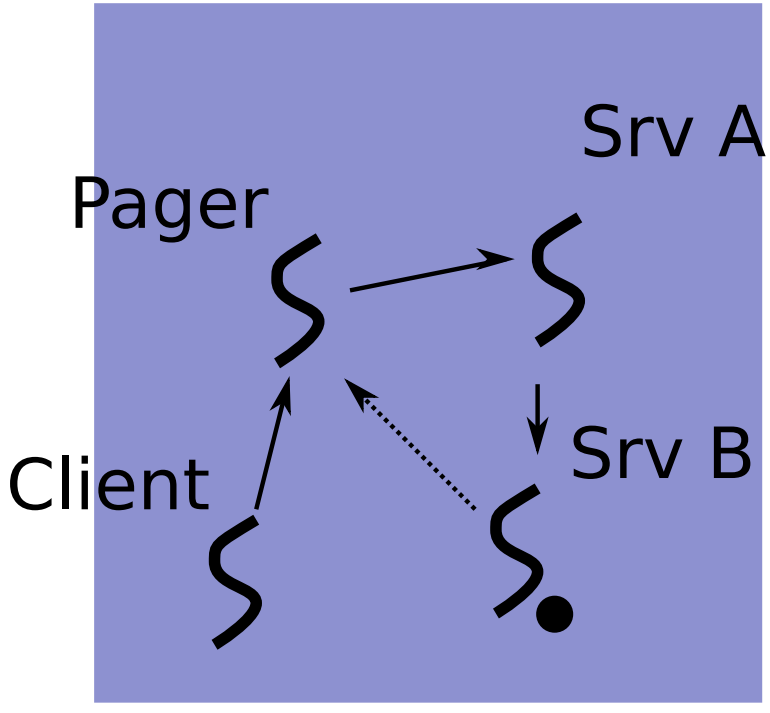
Semphores work cross-CPU. Communication via Semaphores/recall.
“Non-donating” cross-CPU IPC never really needed.
Servers can be CPU-topology aware!



http://os.inf.tu-dresden.de/papers_ps/steinberg_eurosys2010.pdf

06 Livelock

It's possible to construct helping loops... Ouch!



06 Livelock

It's possible to construct helping loops... Ouch!

- Kernel detects loop
- Random IPC is aborted