Core Slicing: Closing the gap between leaky confidential VMs and bare-metal cloud

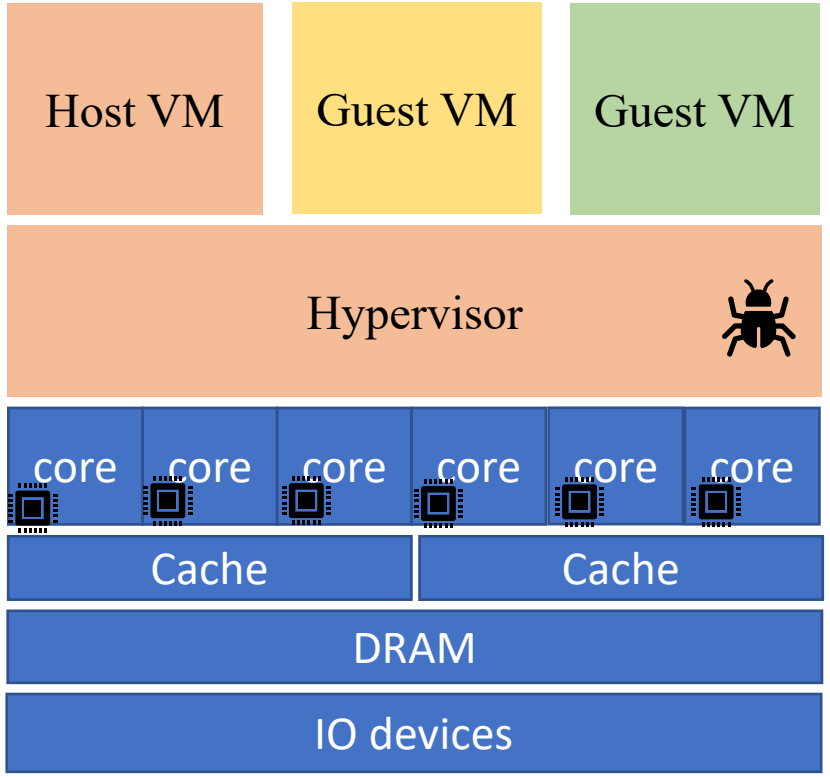
Ziqiao Zhou, Yizhou Shan, Weidong Cui, Xinyang Ge, Marcus Peinado, Andrew Baumann

Microsoft

UC San Diego

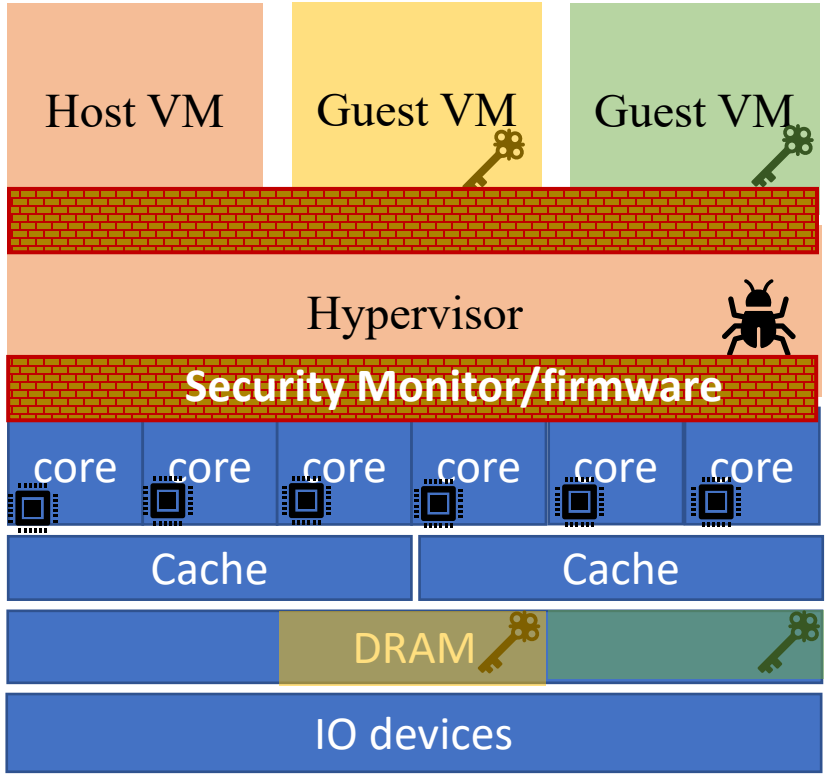
Background: Confidential VMs

- **Goal**: Remove hypervisor from TCB
- **Solution**: Deprivilege hypervisor
- **Examples**:
  - AMD SEV, Intel TDX, Arm CCA



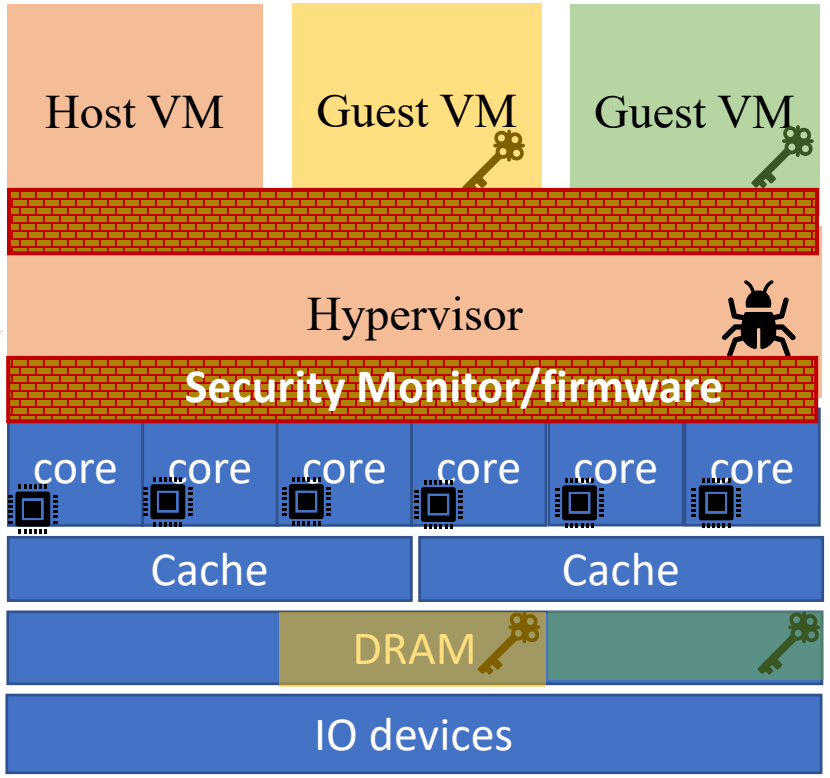
Background: Confidential VMs

- **Goal**: Remove hypervisor from TCB
- **Solution**: Deprivilege hypervisor
- **Examples**:
  - AMD SEV, Intel TDX, Arm CCA



Background: Confidential VMs

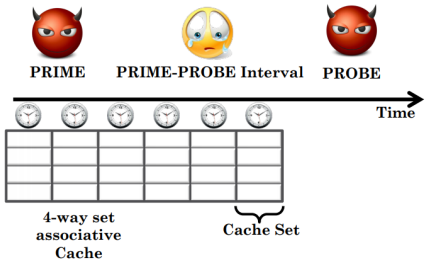
- **Goal**: Remove hypervisor from TCB
- **Solution**: Deprivilege hypervisor
- **Examples**:
  - AMD SEV, Intel TDX, Arm CCA

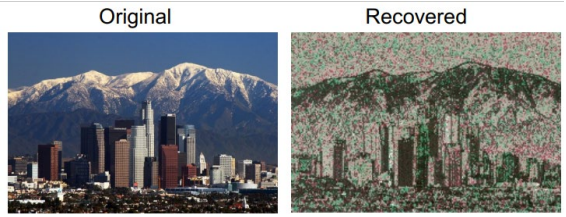


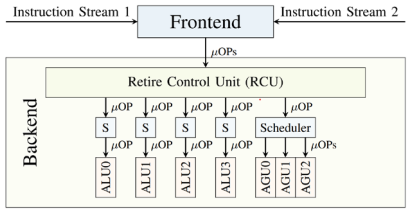
Hypervisor still runs in the same core with VMs

Never-ending side channels

Never-ending side channels

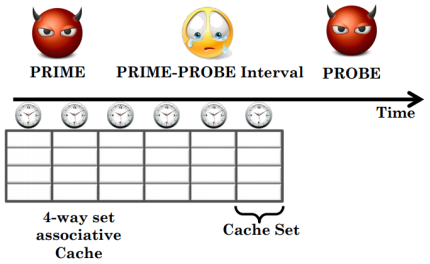


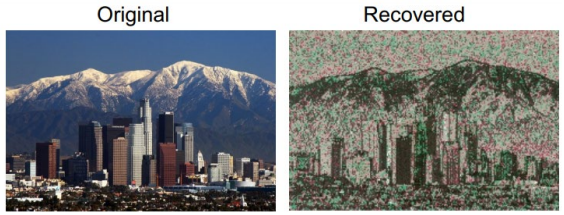


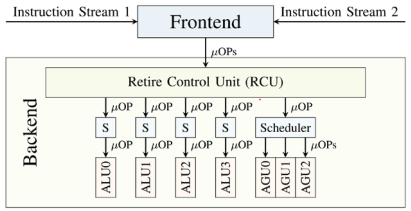


Execution-unit contention (SP’23)

Never-ending side channels



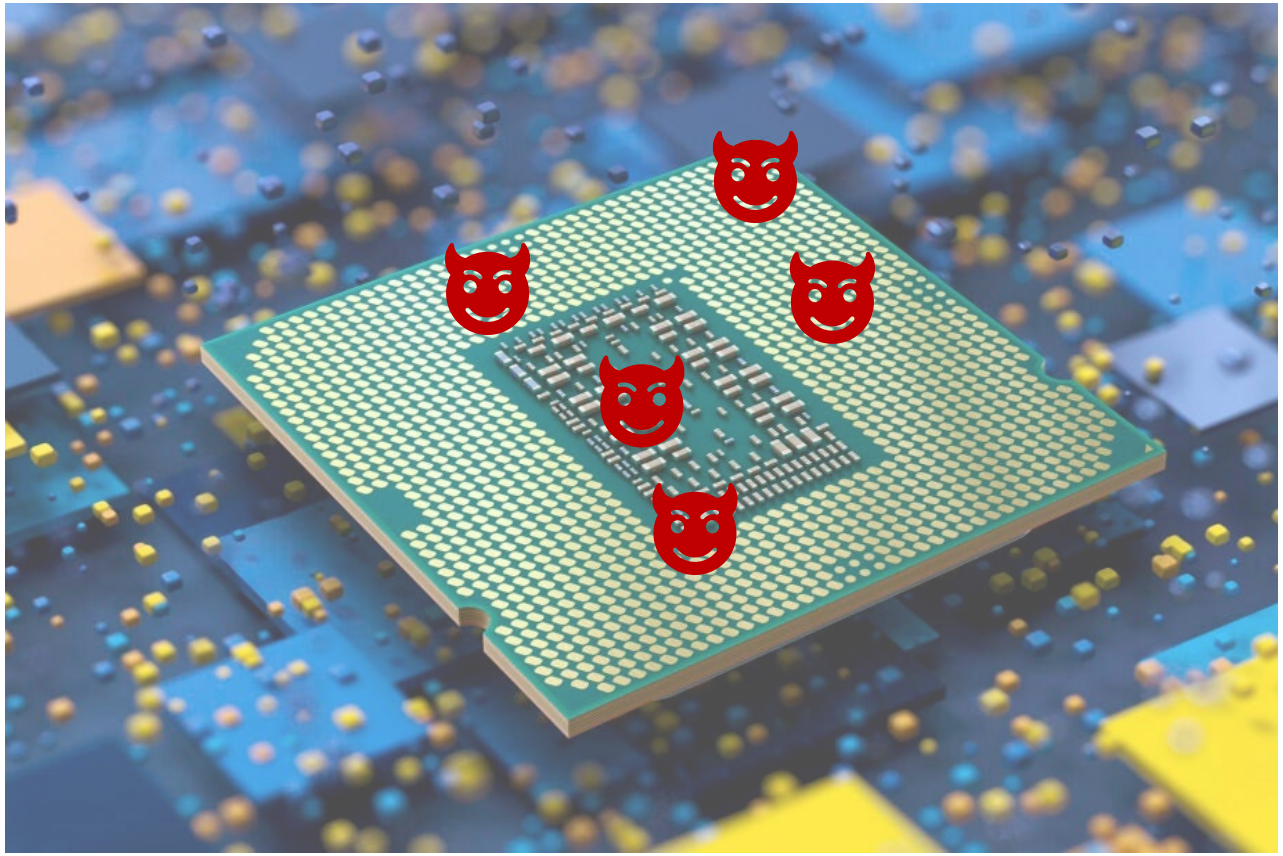




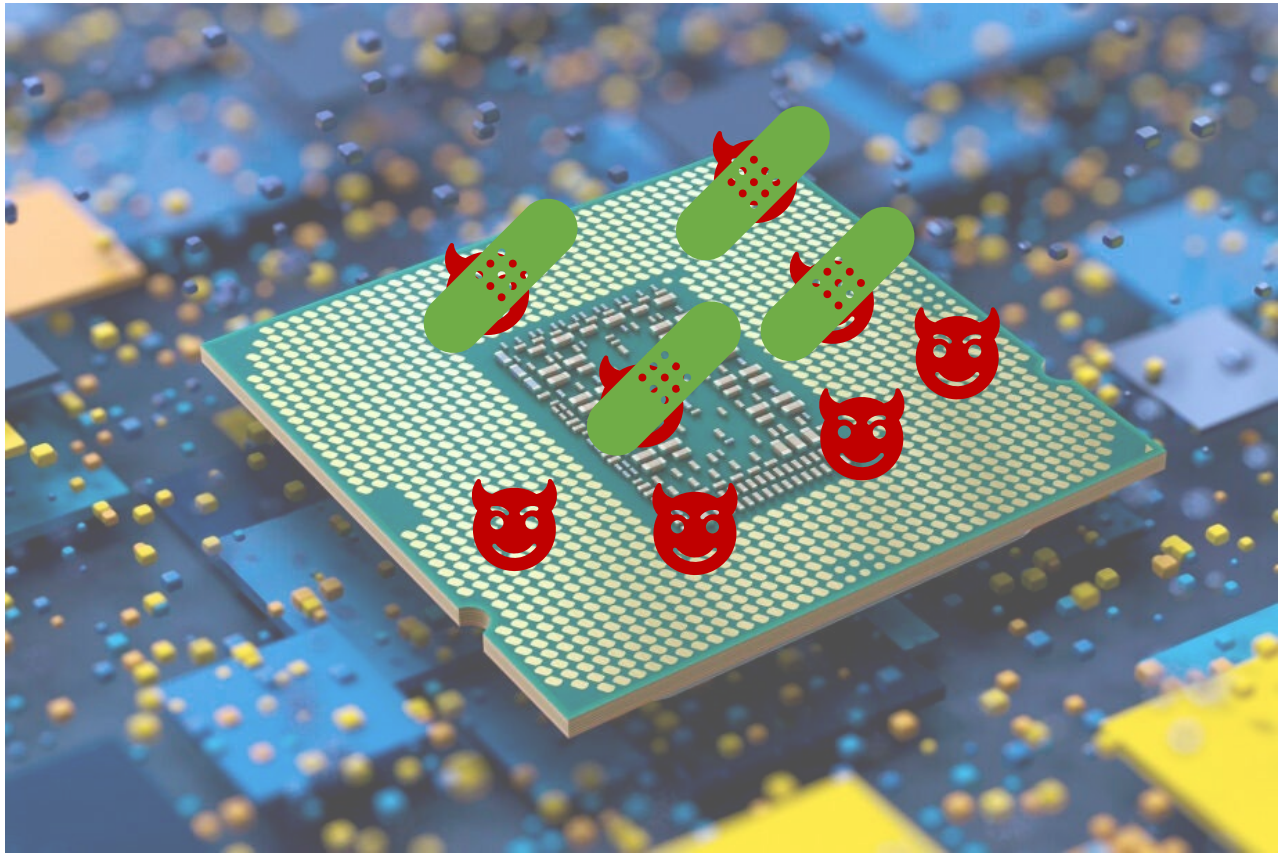
Execution-unit contention (SP'23)

Transient execution attacks	CVE	Intel	AMD	ARM
Meltdown/Supervisor-only bypass	CVE-2017-5754	Y	N	Y
Bound check bypass (Spectre-1)	CVE-2017-5753	Y	Y	Y
Branch Target Injection (Spectre-2)	CVE-2017-5715	Y	Y	Y
Speculative Store Bypass (Spectre-NG-4)	CVE-2018-3639	Y	Y	Y
Rogue System Register Read (Spectre-NG-3a)	CVE-2018-3640	Y	N	Y
Lazy FP State Restore (Spectre-NG)	CVE-2018-3665	Y	N	N
ForeShadow	CVE-2018-3615	Y	M	U
Bounds Check Bypass Store	CVE-2018-3693	Y	Y	Y
Straight-line Speculation	CVE-2021-26341	Y	Y	Y
Return Stack Buffer (Spectre-RSB)	CVE-2022-29901 CVE-2022-23824	Y	Y	Y
Speculative Vectorization Exploits (Spectre-HD)	(2023-02)	Y	Y	Y
…				

Confidential VMs: **Reactive** mitigations



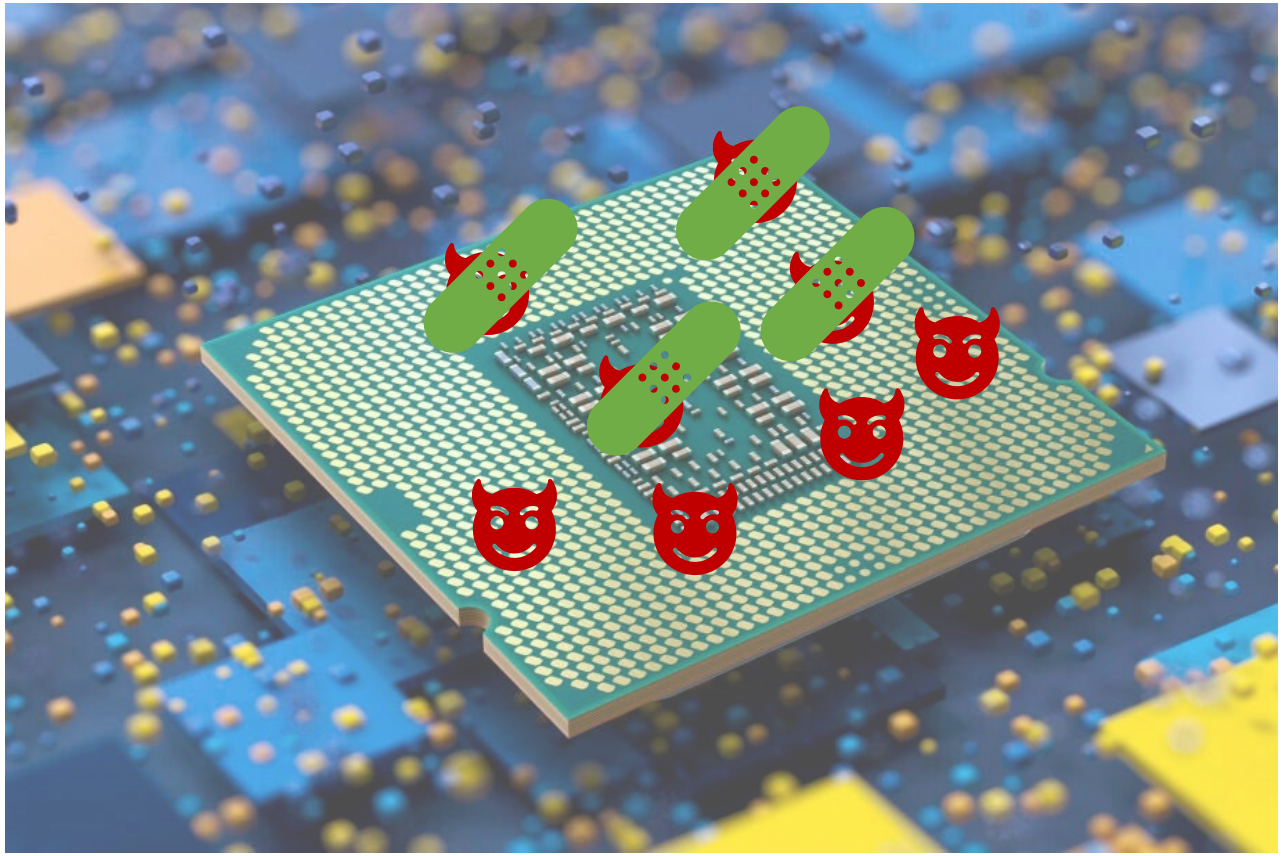
Confidential VMs: **Reactive** mitigations



Confidential VMs: **Reactive** mitigations

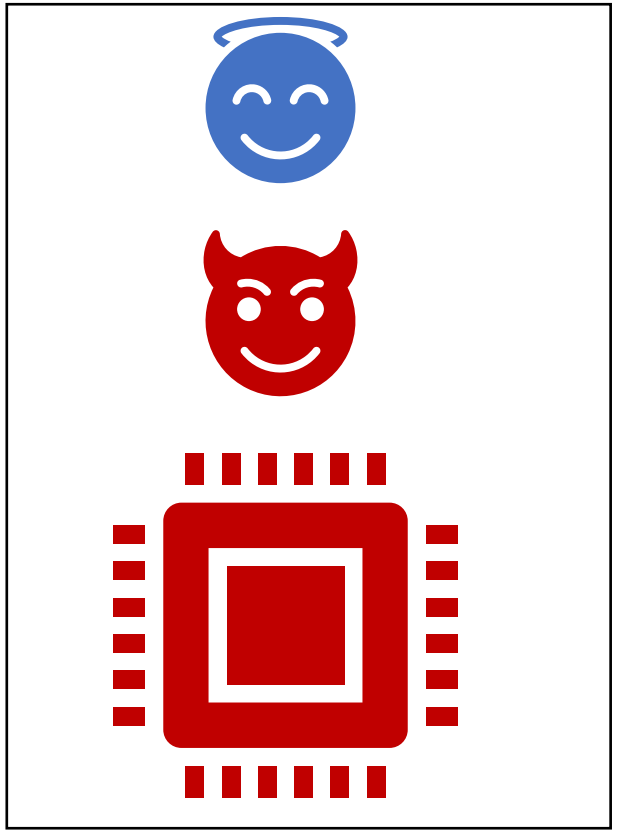
"Intel believes removing all incidental channels from computing systems is not in customer's best interests and is not feasible, nor is it feasible to completely prevent the intentional misuse of incidental channels. "

-Intel



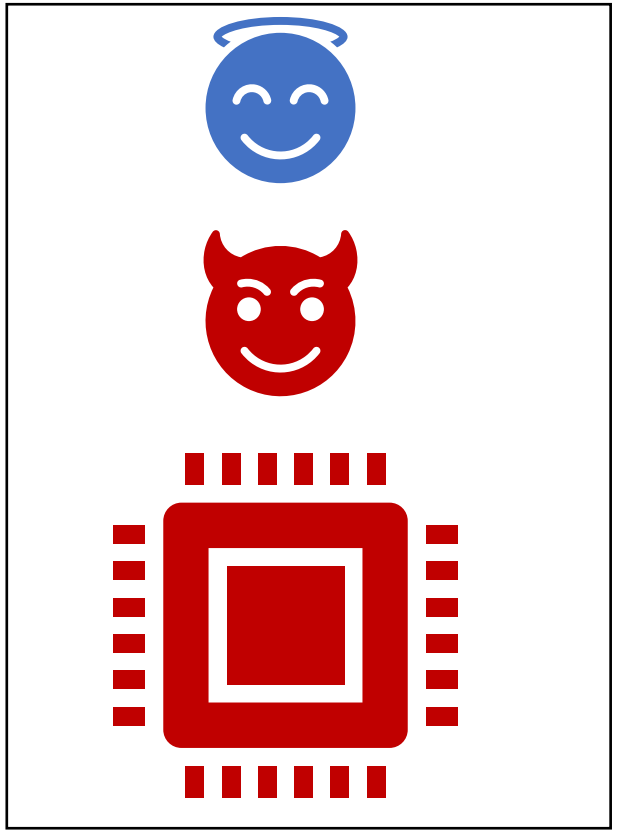
How to eliminate side-channel attacks

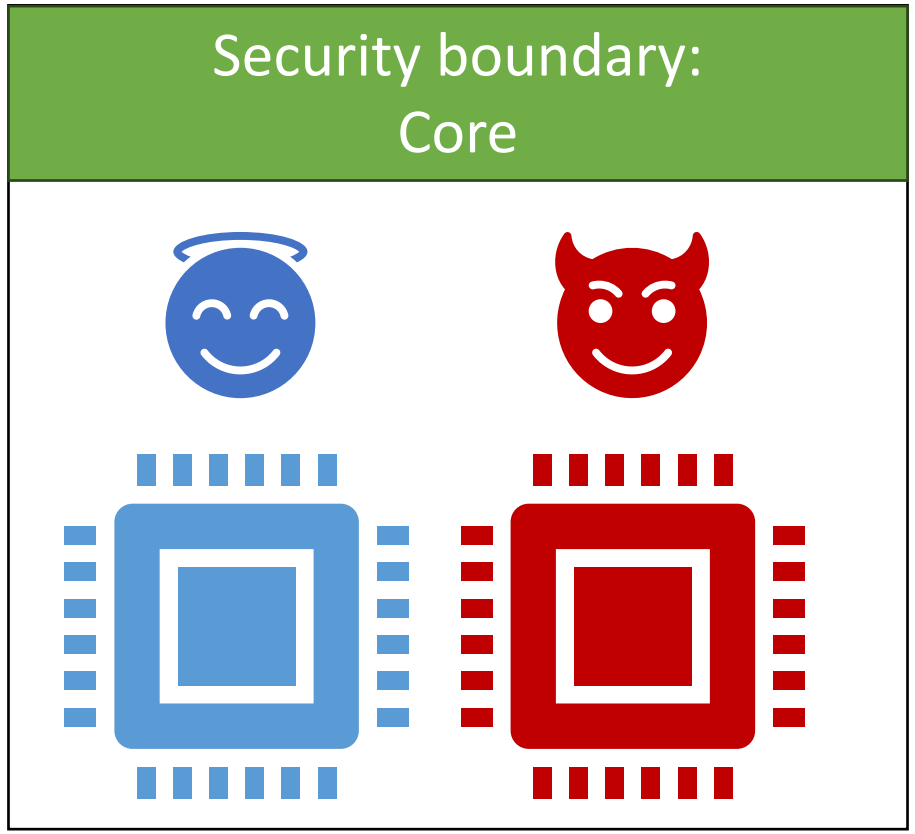
- Adversary-controlled code in the same core



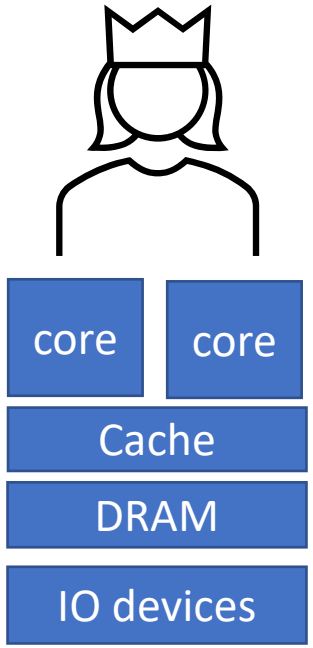
How to eliminate side-channel attacks

- Adversary-controlled code in the same core



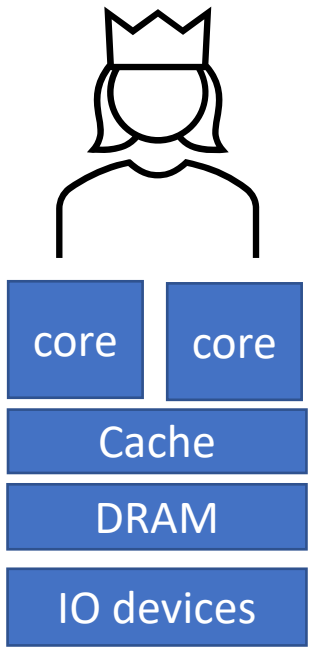


Existing solution: bare-metal cloud



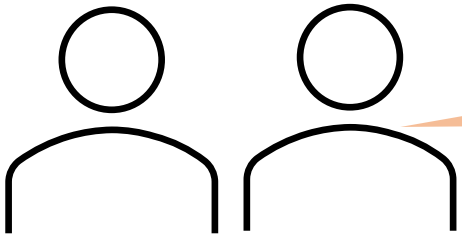
Existing solution: bare-metal cloud

- **Pros**: Strong isolation, predicable performance
- **Cons**: Lack of flexibility

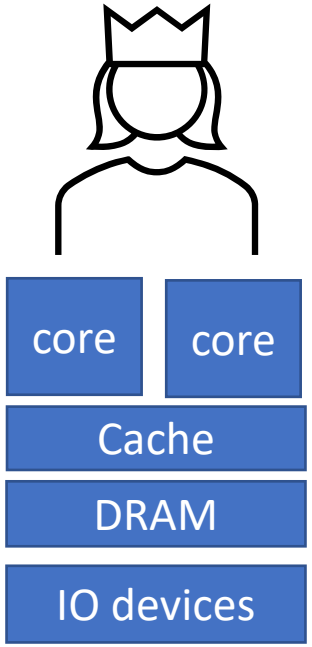


Existing solution: bare-metal cloud

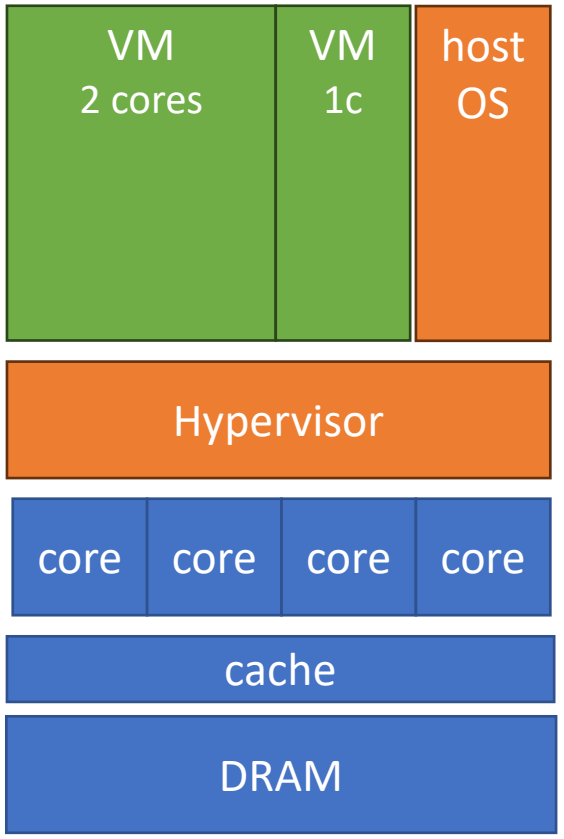
- **Pros**: Strong isolation, predicable performance
- **Cons**: Lack of flexibility



We want “bare-metal” security and performance, but at sub-machine granularity.



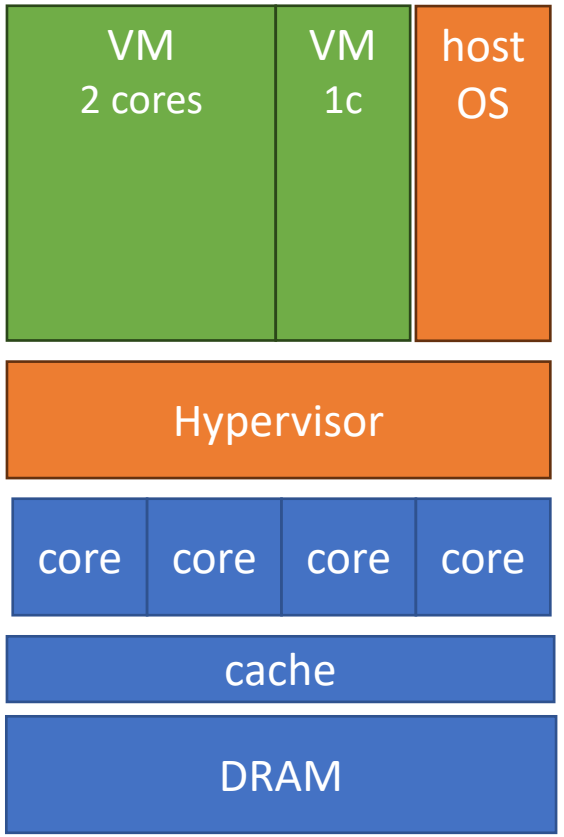
Hypervisor in today's IaaS cloud



Hypervisor in today's IaaS cloud

Resources sold should match those available.

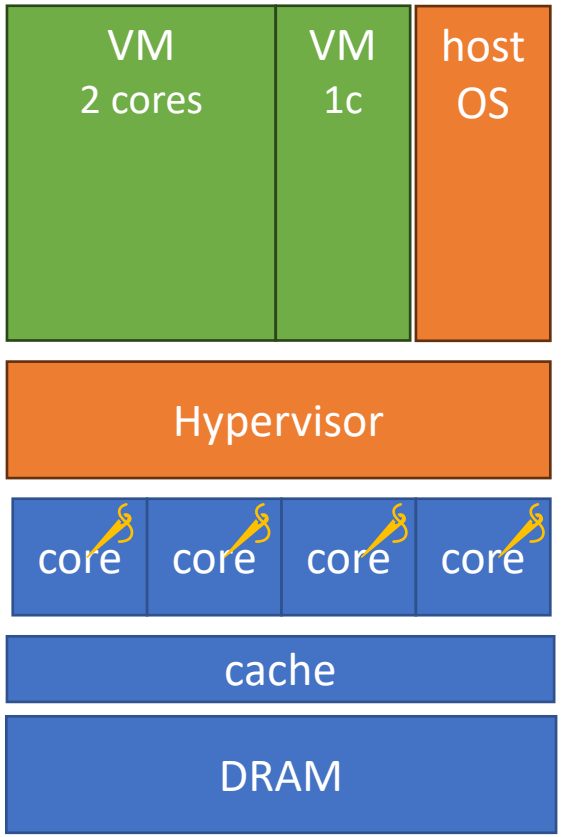
- Discrete cores
  - No time slicing
- Static memory
  - No ballooning or demand paging
- I/O offload



Hypervisor in today's IaaS cloud

Resources sold should match those available.

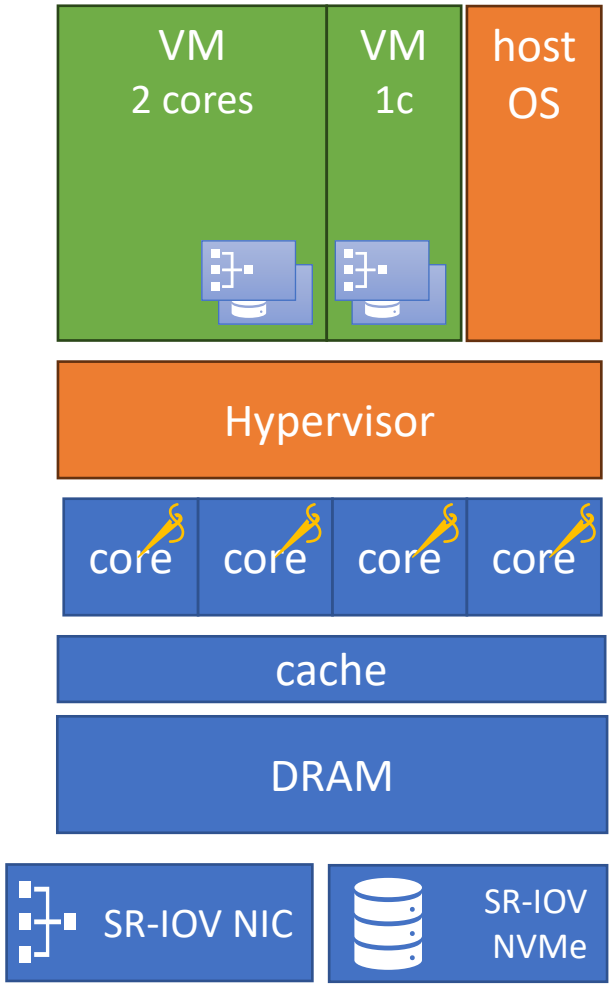
- Discrete cores
  - No time slicing
- Static memory
  - No ballooning or demand paging
- I/O offload



Hypervisor in today's IaaS cloud

Resources sold should match those available.

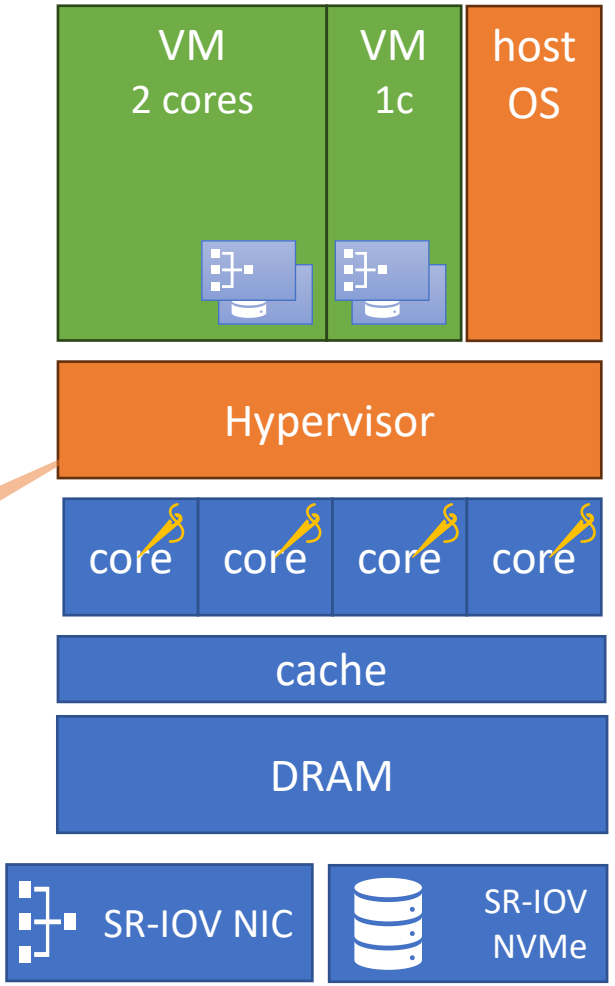
- Discrete cores
  - No time slicing
- Static memory
  - No ballooning or demand paging
- I/O offload



Hypervisor in today's IaaS cloud

Resources sold should match those available.

- Discrete cores
  - No time slicing
- Static memory
  - No ballooning or demand paging
- I/O offload



Hypervisor is only used for isolation.

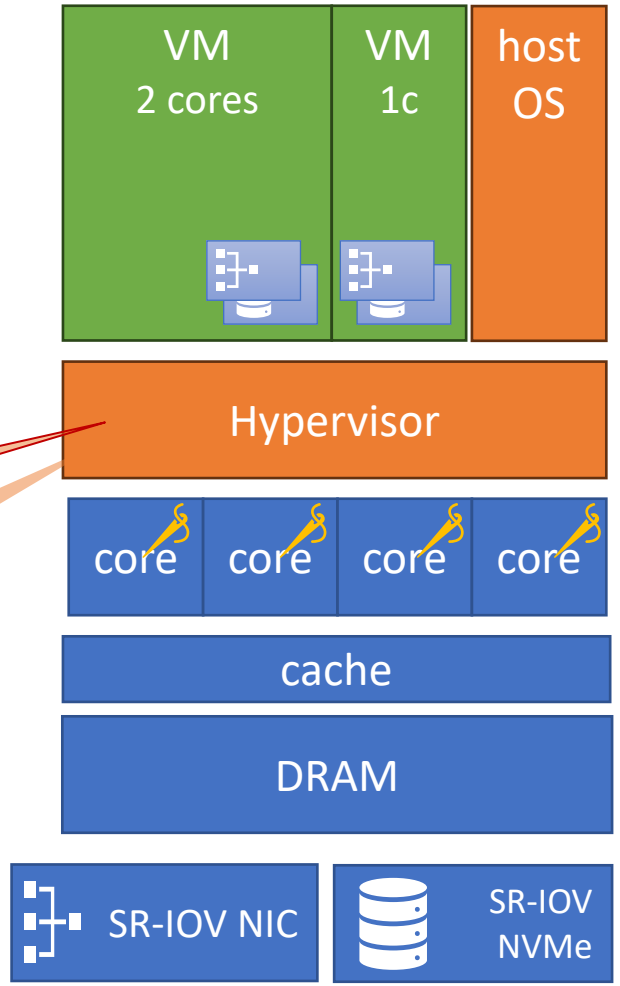
Hypervisor in today's IaaS cloud

Resources sold should match those available.

- Discrete cores
  - No time slicing
- Static memory
  - No ballooning or demand paging
- I/O offload

Hypervisor is the source of CPU sharing.

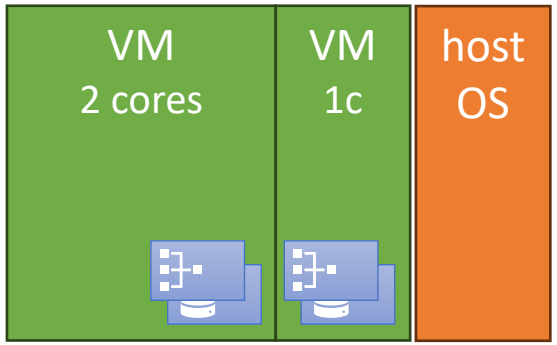
Hypervisor is only used for isolation.

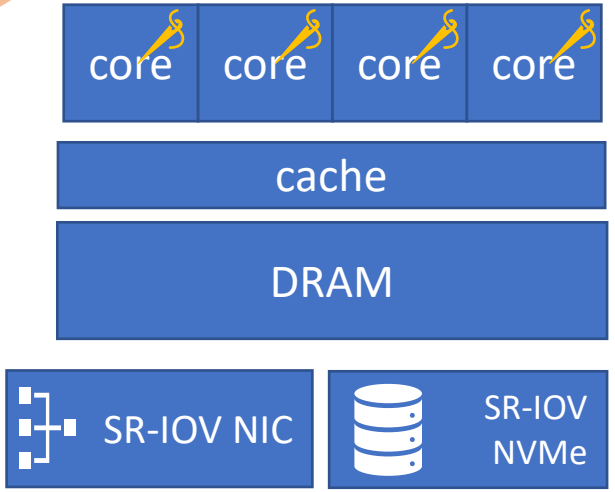


Hypervisor in today's IaaS cloud

Resources sold should match those available.

- Discrete cores
  - No time slicing
- Static memory
  - No ballooning or demand paging
- I/O offload



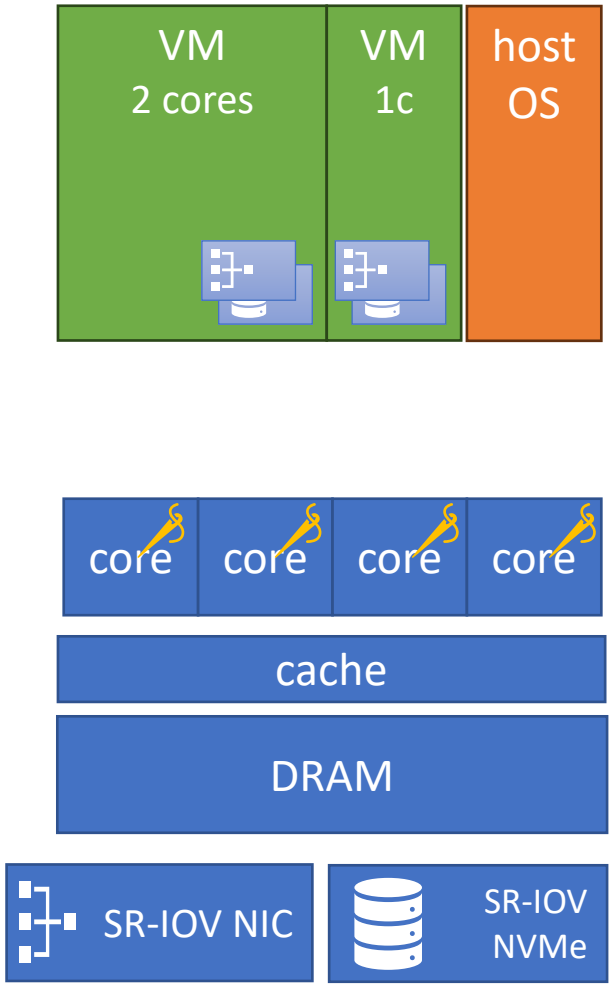


Hypervisor is the source of CPU sharing.

Hypervisor is only used for isolation.

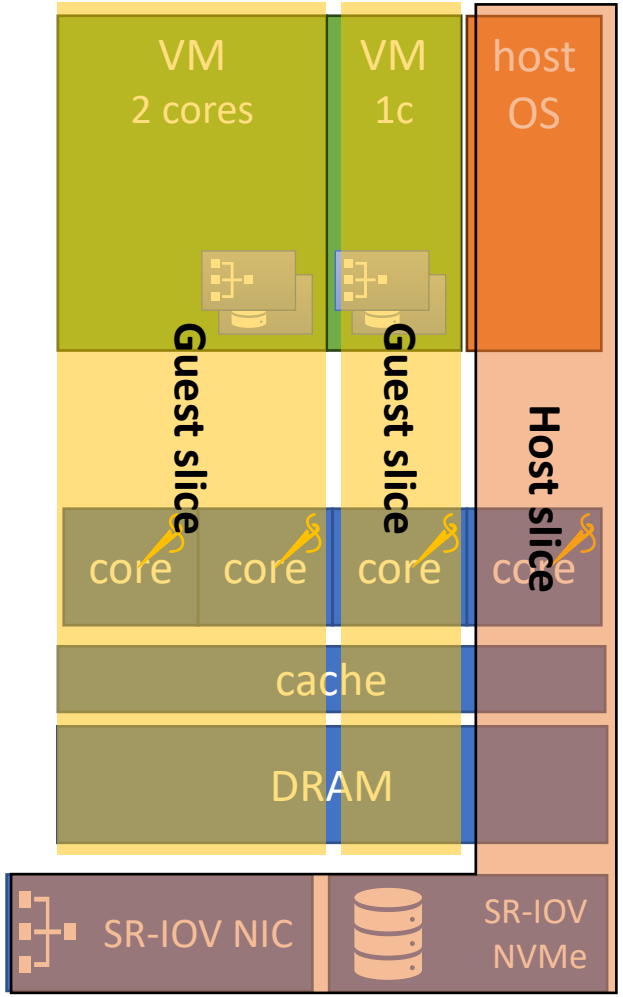
Idea of core slicing

- ✓ **Each slice gets a physical partition**
  - Exclusive CPU
    - No CPU virtualization layer
  - Exclusive DRAM partition
    - No additional memory translation
  - Directly access dedicated I/O devices
    - e.g., access virtual devices via SR-IOV



Idea of core slicing

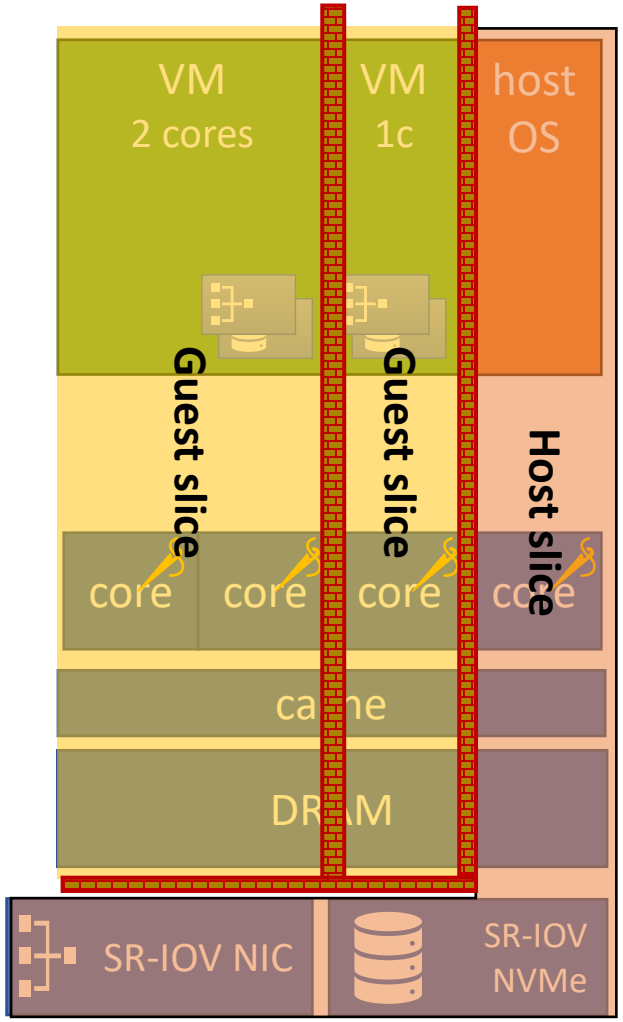
- ✓ **Each slice gets a physical partition**
  - Exclusive CPU
    - No CPU virtualization layer
  - Exclusive DRAM partition
    - No additional memory translation
  - Directly access dedicated I/O devices
    - e.g., access virtual devices via SR-IOV



Idea of core slicing

- ✓ **Each slice gets a physical partition**
  - Exclusive CPU
    - No CPU virtualization layer
  - Exclusive DRAM partition
    - No additional memory translation
  - Directly access dedicated I/O devices
    - e.g., access virtual devices via SR-IOV

- ✓ **Hardware-assisted isolation**



Two lightweight hardware features

- **Per-core lockable filter** registers
  - Defines hardware resources (e.g., DRAM)
  - Locked until a secure reset

- A secure **per-core reset** unit
  - Clear per-core state

Two lightweight hardware features

- **Per-core lockable filter** registers
  - Defines hardware resources (e.g., DRAM)
  - Locked until a secure reset

- A secure **per-core reset** unit
  - Clear per-core state

Two lightweight hardware features

- **Per-core lockable filter** registers
  - Defines hardware resources (e.g., DRAM)
  - Locked until a secure reset

- A secure **per-core reset** unit
  - Clear per-core state

Lockable filter registers: Slicing DRAM

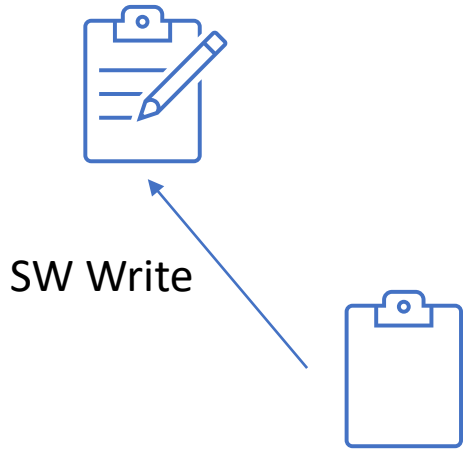
- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges

Lockable filter registers: Slicing DRAM

- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges

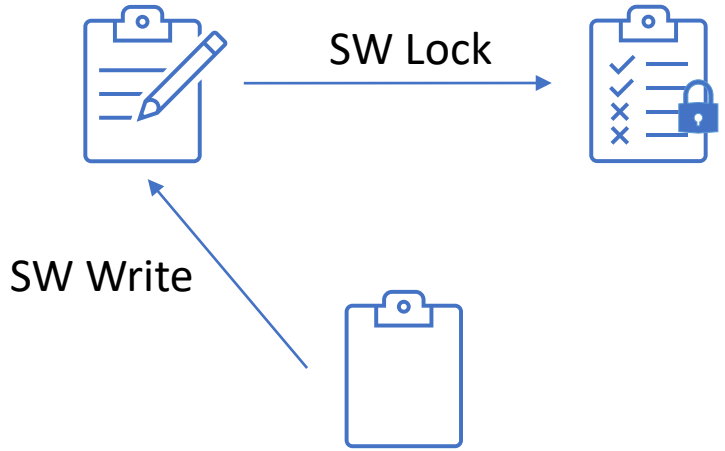
Lockable filter registers: Slicing DRAM

- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges



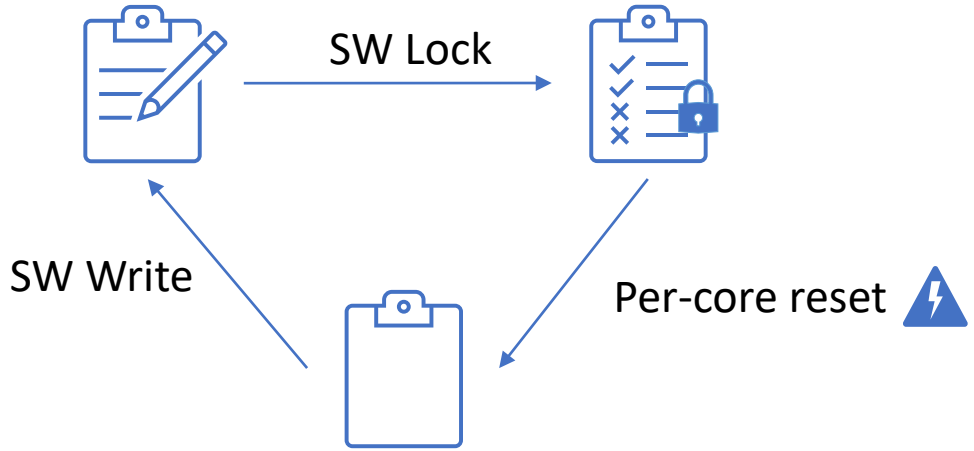
Lockable filter registers: Slicing DRAM

- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges



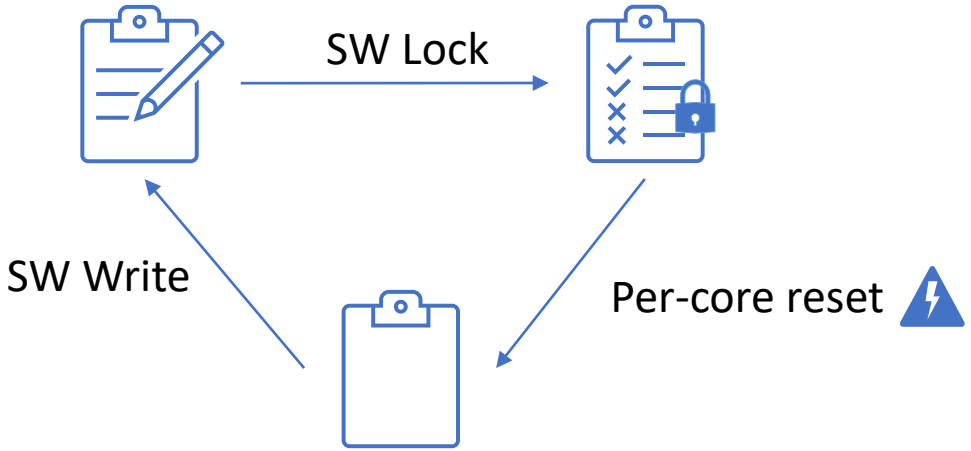
Lockable filter registers: Slicing DRAM

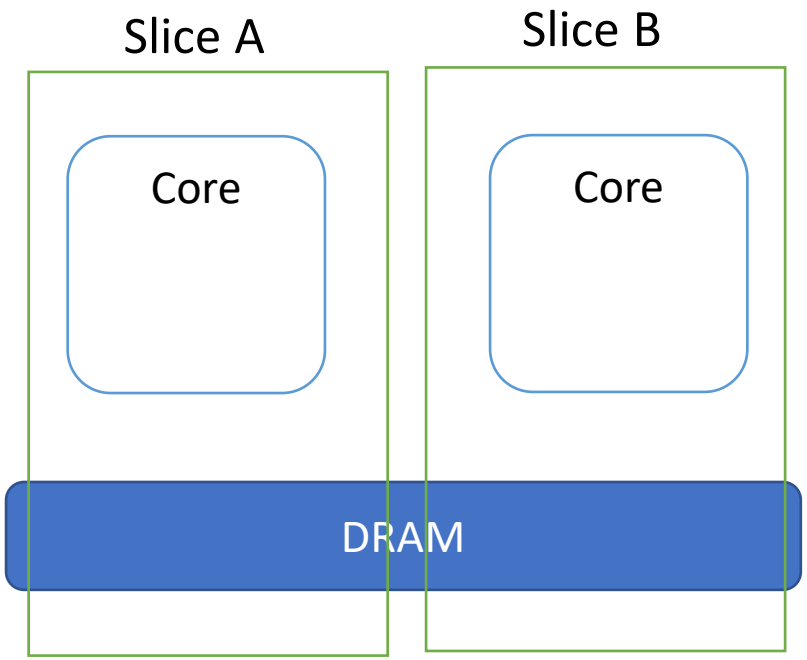
- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges



Lockable filter registers: Slicing DRAM

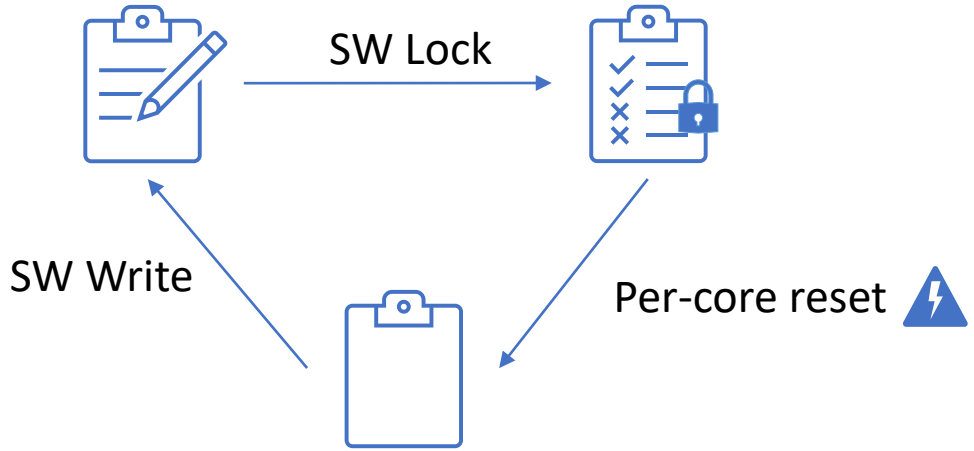
- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges

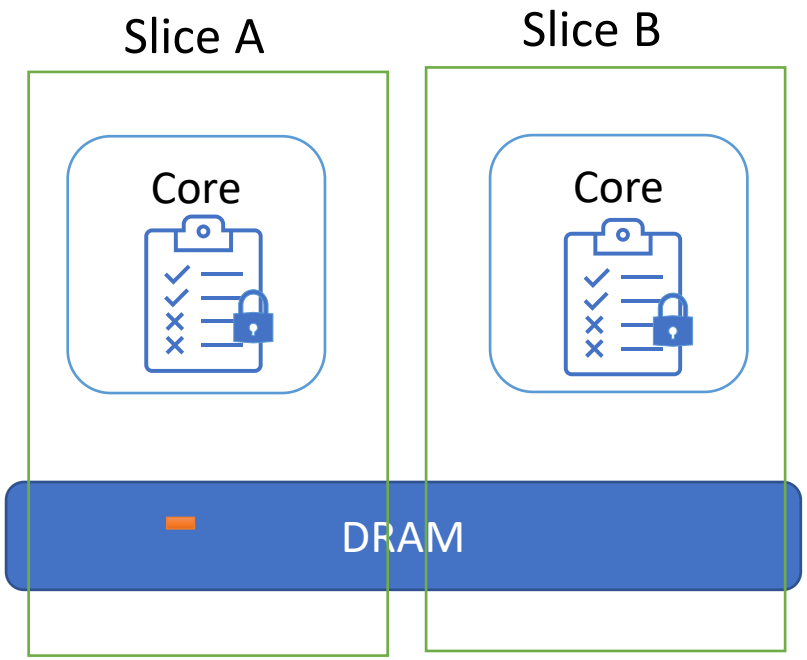




Lockable filter registers: Slicing DRAM

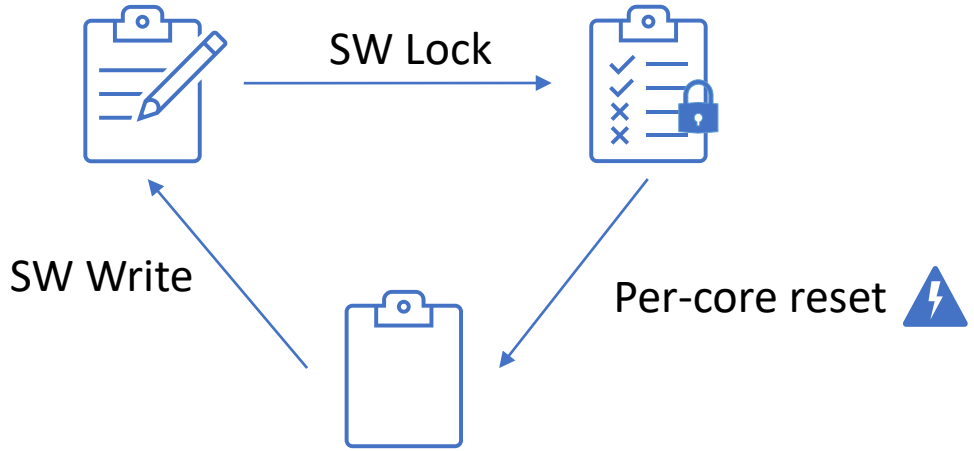
- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges

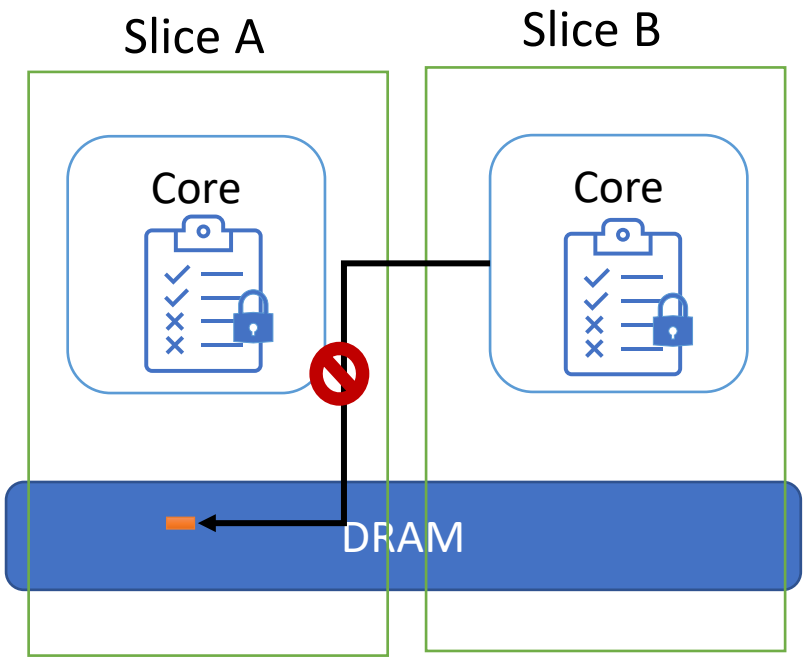




Lockable filter registers: Slicing DRAM

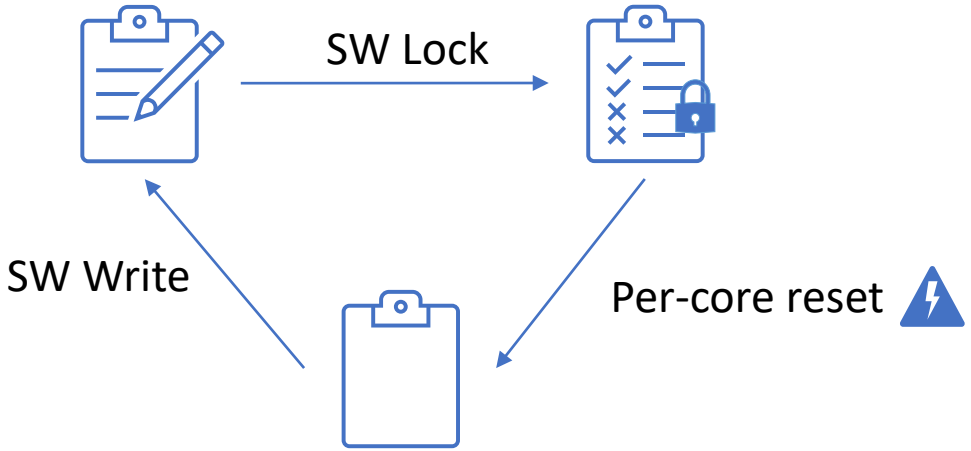
- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges

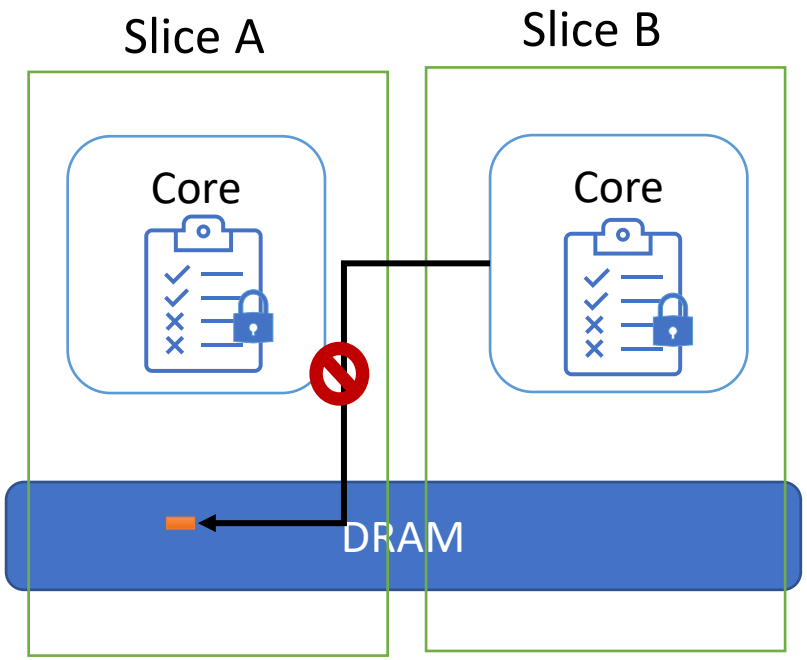




Lockable filter registers: Slicing DRAM

- Per-core lockable filter registers
  - Allowlist and denylist for DRAM ranges

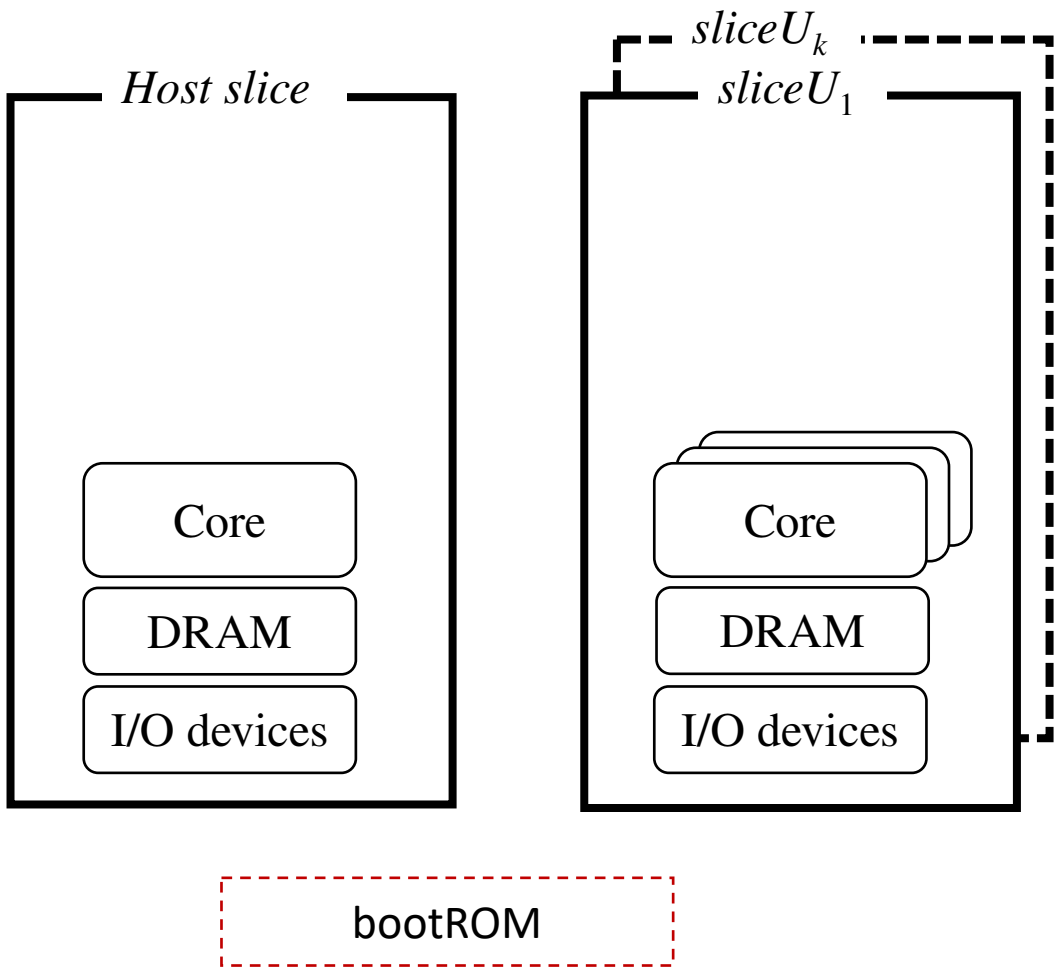




Slicing other resources

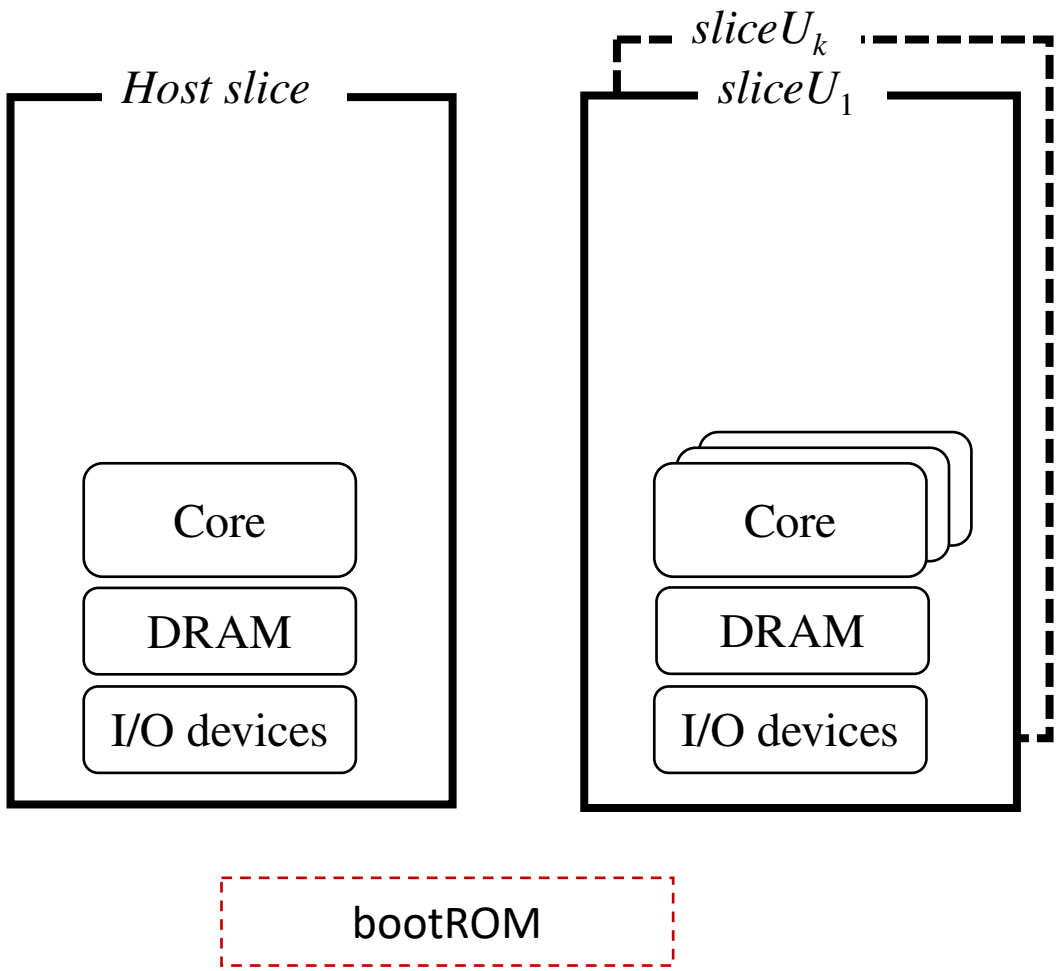
- Interrupts
- I/O devices
- Cache
  - SiFive: way masking for the shared cache.
  - Intel: Cache Allocation Technology (CAT).
- DMA
  - RISC-V: IOPMP
  - X86: IOMMU

Core slicing software stack



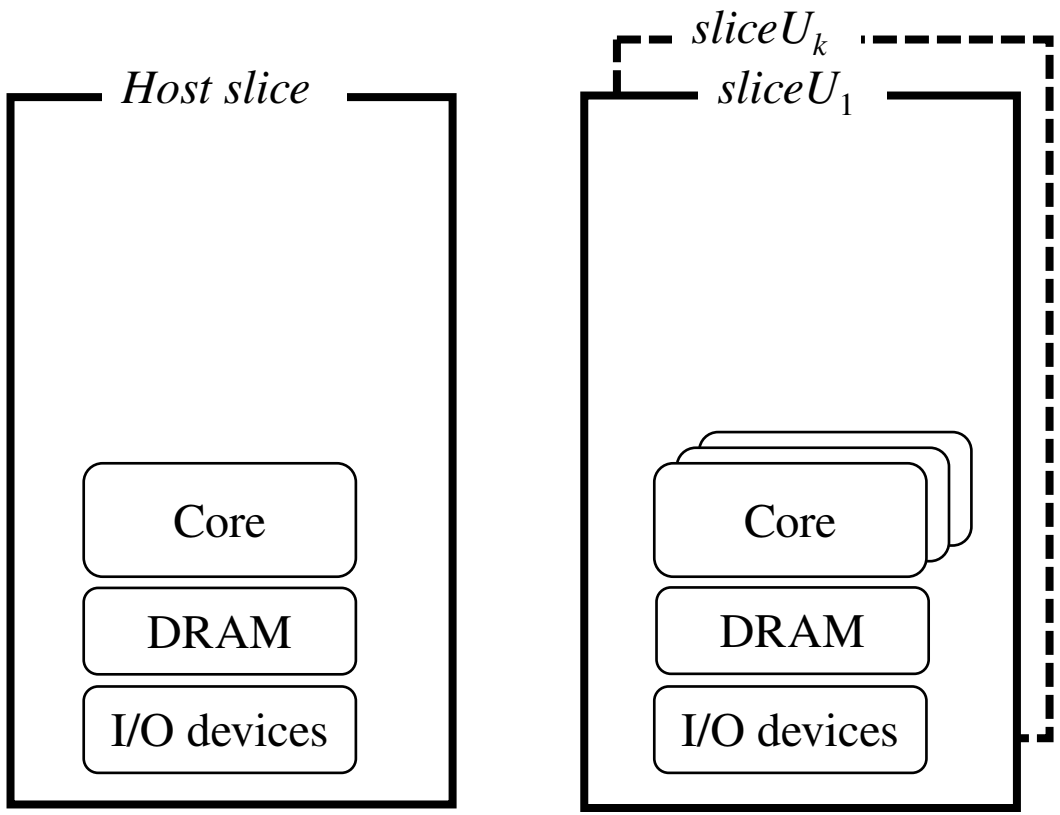
Core slicing software stack

- Bootloader (sliceloader) is **trusted**
- Slicevisor is trusted only at control path
  - Slice creation
  - Slice destroy.
- Untrusted firmware in guest slices



Core slicing software stack

- Bootloader (sliceloader) is **trusted**
- Slicevisor is trusted only at control path
  - Slice creation
  - Slice destroy.
- Untrusted firmware in guest slices

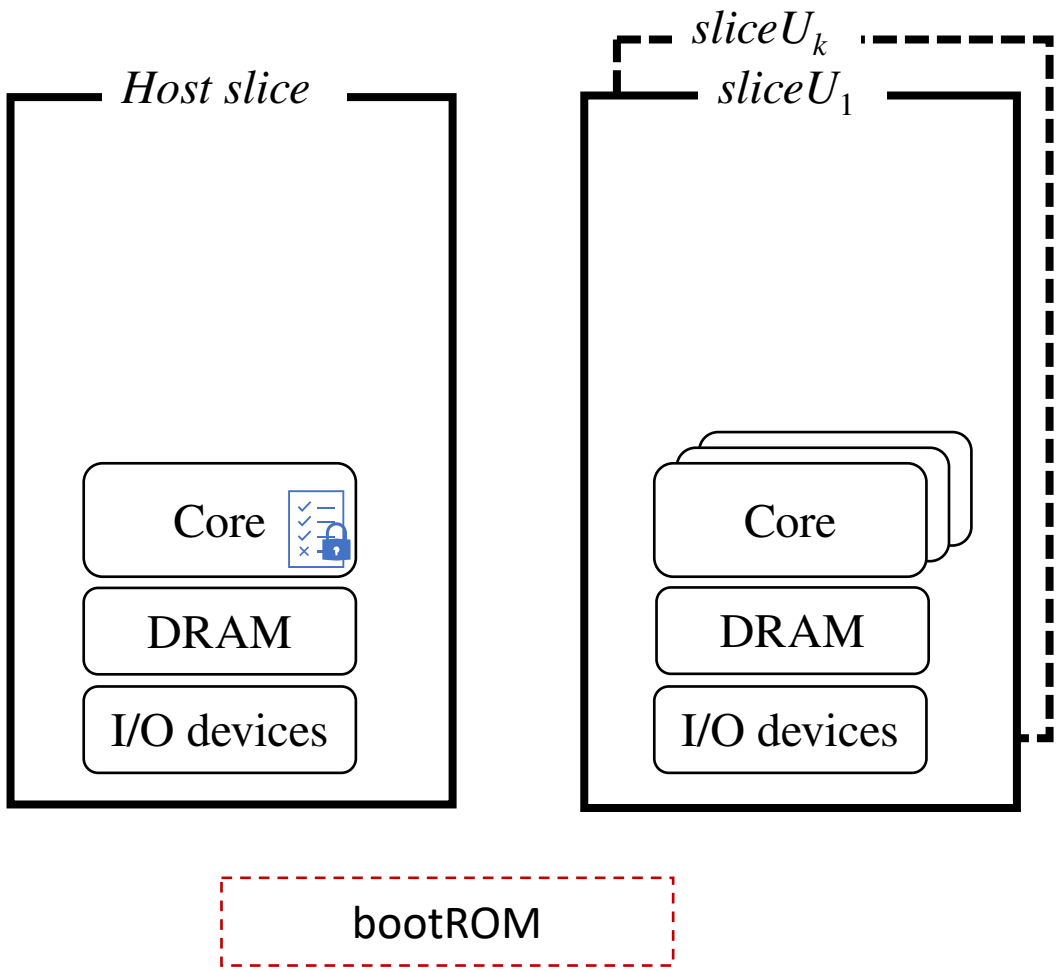


bootROM

Read a protected slice configuration to determine and validate the role of the core

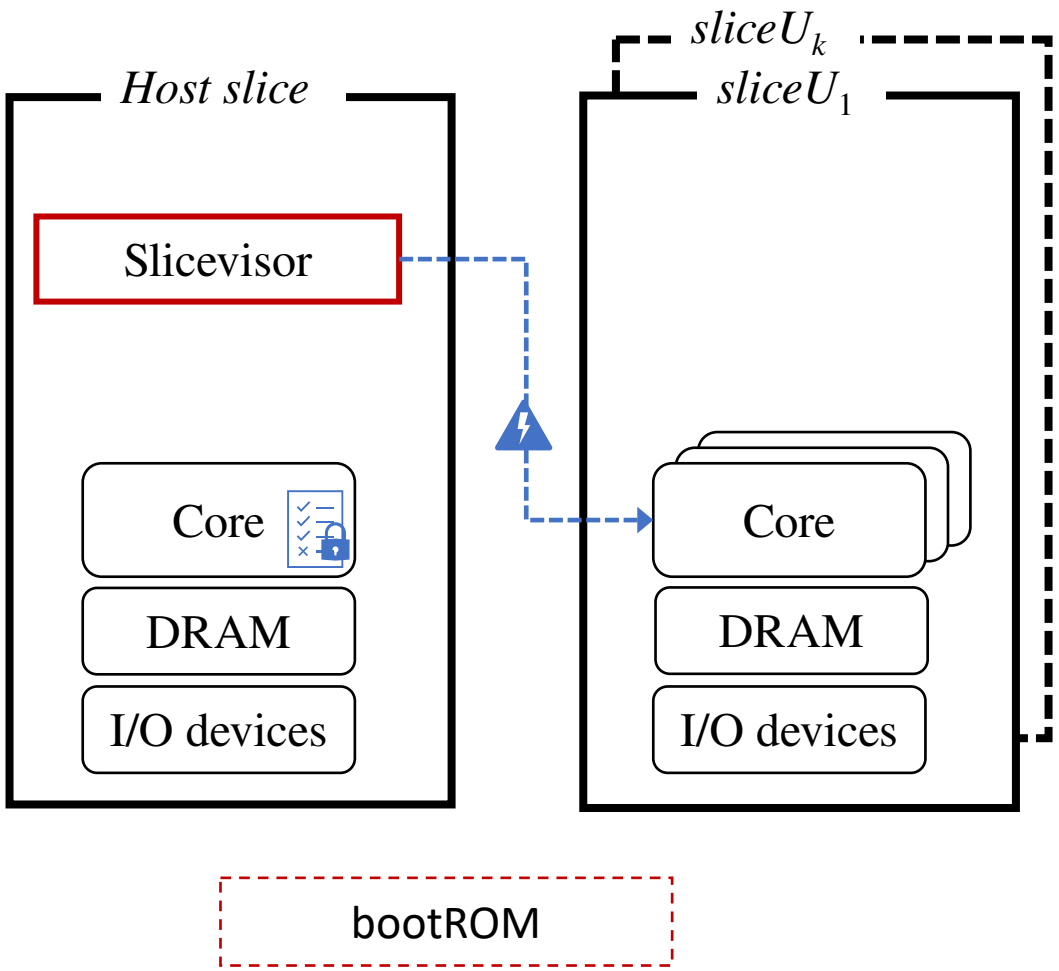
Core slicing software stack

- Bootloader (sliceloader) is **trusted**
- Slicevisor is trusted only at control path
  - Slice creation
  - Slice destroy.
- Untrusted firmware in guest slices



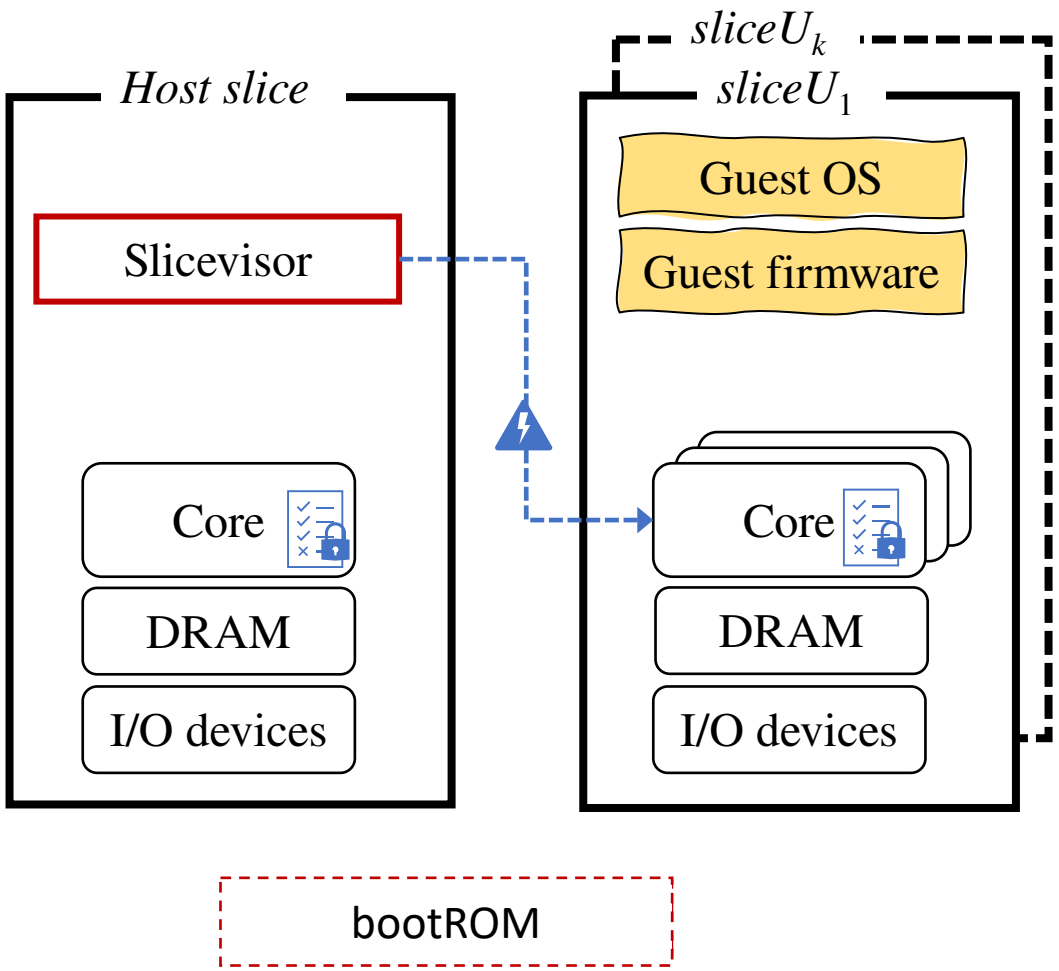
Core slicing software stack

- Bootloader (sliceloader) is **trusted**
- Slicevisor is trusted only at control path
  - Slice creation
  - Slice destroy.
- Untrusted firmware in guest slices



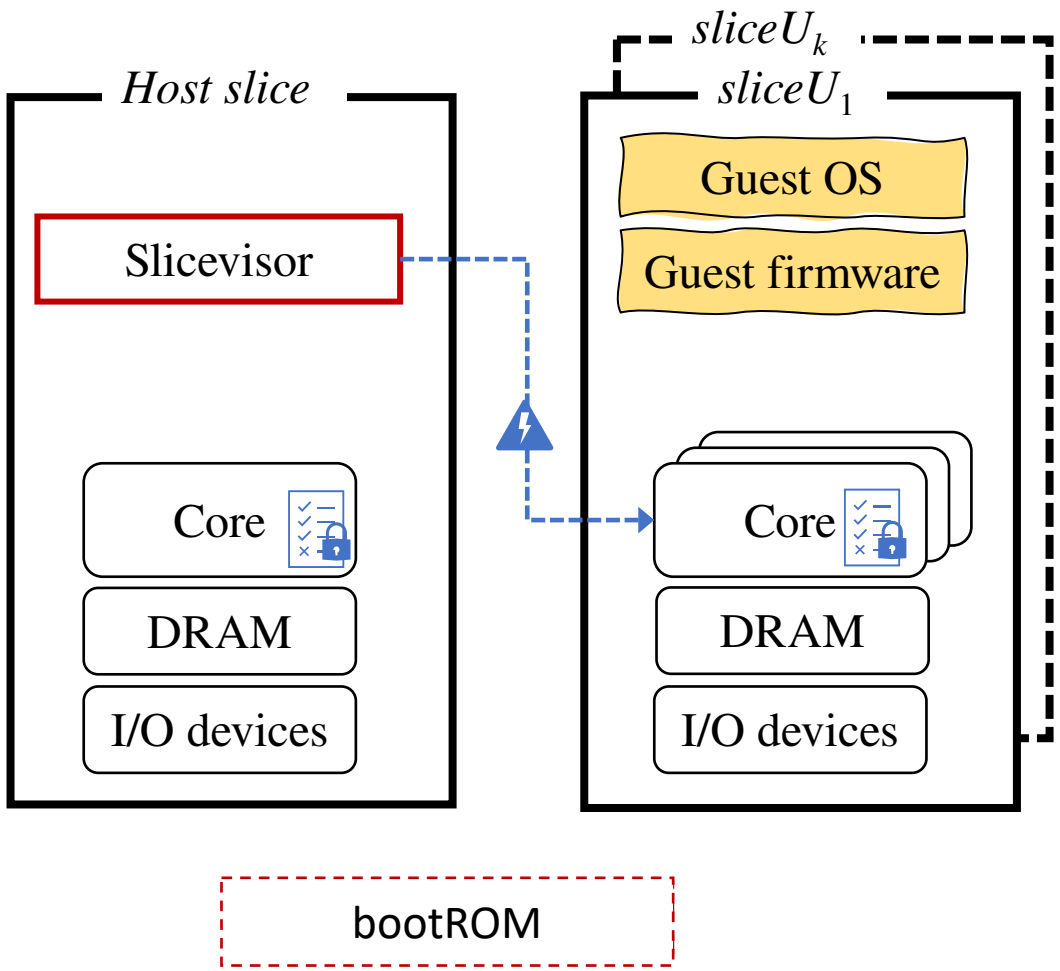
Core slicing software stack

- Bootloader (sliceloader) is **trusted**
- Slicevisor is trusted only at control path
  - Slice creation
  - Slice destroy.
- Untrusted firmware in guest slices



Core slicing software stack

- Bootloader (sliceloader) is **trusted**
- Slicevisor is trusted only at control path
  - Slice creation
  - Slice destroy.
- Untrusted firmware in guest slices



No trusted and shared software cross slices

Our prototype in RISC-V

PMP for

- DRAM isolation
- Memory-mapped
  - MSRs
  - IO
  - interrupts

```
verification success.
hart_count = 2
mem_size = 20000000
digest:
ac08e3f644174b86e10284fca26aba368b79d89404342c9f80b135daa829a7616e546357

>> slice help
[72.740933] slice_help(): slice STOP -- stop a slice.
[72.744335] slice_help(): slice START -- start a slice.
[72.745143] slice_help(): slice CREATE -- create a slice.
[72.745744] slice_help(): slice DELETE -- delete a slice.
[72.746607] slice_help(): slice ATTEST -- attest a slice.
```

```
Jan  1 00:00:02 login[83]: root login on 'con'
~ # more /proc/cpuinfo
processor       : 0
hart            : 1
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0

processor       : 1
hart            : 2
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0
```

```
Jan  1 00:00:05 login[90]: root login on 'con
~ # cat /proc/cpuinfo
processor       : 0
hart            : 3
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0

processor       : 1
hart            : 4
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0
```

Our prototype in RISC-V

PMP for

- DRAM isolation
- Memory-mapped
  - MSRs
  - IO
  - interrupts

```
verification success.
hart_count = 2
mem_size = 20000000
digest:
ac08e3f644174b86e10284fca26aba368b79d89404342c9f80b135daa829a7616e546357

>> slice help
[72.740933] slice_help(): slice STOP -- stop a slice.
[72.744335] slice_help(): slice START -- start a slice.
[72.745143] slice_help(): slice CREATE -- create a slice.
[72.745744] slice_help(): slice DELETE -- delete a slice.
[72.746607] slice_help(): slice ATTEST -- attest a slice.
```

```
Jan  1 00:00:02 login[83]: root login on 'con'
~ # more /proc/cpuinfo
processor       : 0
hart            : 1
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0

processor       : 1
hart            : 2
isa             : rv64imafdc
mmu             : sv39
```

```
Jan  1 00:00:05 login[90]: root login on 'con
~ # cat /proc/cpuinfo
processor       : 0
hart            : 3
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0

processor       : 1
hart            : 4
isa             : rv64imafdc
mmu             : sv39
```

Our prototype in RISC-V

PMP for

- DRAM isolation
- Memory-mapped
  - MSRs
  - IO
  - interrupts

Exclusive resources

```
verification success.
hart_count = 2
mem_size = 20000000
digest:
ac08e3f644174b86e10284fca26aba368b79d89404342c9f80b135daa829a7616e546357

>> slice help
[72.740933] slice_help(): slice STOP -- stop a slice.
[72.744335] slice_help(): slice START -- start a slice.
[72.745143] slice_help(): slice CREATE -- create a slice.
[72.745744] slice_help(): slice DELETE -- delete a slice.
[72.746607] slice_help(): slice ATTEST -- attest a slice.
```

```
Jan  1 00:00:02 login[83]: root login on 'con'
~ # more /proc/cpuinfo
processor       : 0
hart            : 1
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0

processor       : 1
hart            : 2
isa             : rv64imafdc
mmu             : sv39
```

```
Jan  1 00:00:05 login[90]: root login on 'con
~ # cat /proc/cpuinfo
processor       : 0
hart            : 3
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0

processor       : 1
hart            : 4
isa             : rv64imafdc
mmu             : sv39
```

Our prototype in RISC-V

PMP for
- DRAM isolation
- Memory-mapped
  - MSRs
  - IO
  - interrupts

```
verification success.
hart_count = 2
mem_size = 20000000
digest:
ac08e3f644174b86e10284fca26aba368b79d89404342c9f80b135daa829a7616e546357

>> slice help
[72.740933] slice_help(): slice STOP -- stop a slice.
[72.744335] slice_help(): slice START -- start a slice.
[72.745143] slice_help(): slice CREATE -- create a slice.
[72.745744] slice_help(): slice DELETE -- delete a slice.
[72.746607] slice_help(): slice ATTEST -- attest a slice.
```

```
Jan  1 00:00:02 login[83]: root login on 'con'
~ # more /proc/cpuinfo
processor       : 0
hart            : 1
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0

processor       : 1
hart            : 2
isa             : rv64imafdc
mmu             : sv39
```

```
Jan  1 00:00:05 login[90]: root login on 'con
~ # cat /proc/cpuinfo
processor       : 0
hart            : 3
isa             : rv64imafdc
mmu             : sv39
uarch           : sifive,rocket0

processor       : 1
hart            : 4
isa             : rv64imafdc
mmu             : sv39
```

Exclusive resources

Bare-metal performance

Summary

- TEE that avoids side channels by design
  - No virtualization overhead
  - Resources partitioned at core granularity
- Two lightweight hardware features
  - Lockable filter registers
  - Per-core reset
- More details in the paper, including:
  - Attestation and memory encryption
  - Extending the design beyond RISC-V
  - Evaluation of limited registers and bare-metal performance

Artifact available at https://github.com/msrssp/core-slicing