CVM Extension: Confidential Virtual Machine Extension for RISC-V

Dingji Li

IPADS, SJTU

MULTIPLE C-MODE ID

Infinite Isolated Environment

- **Goal: each CVM should be isolated from all other software**
  - Last meeting: prefer infinite modes with isolation
    - Each CVM resides in its own isolated mode
- **CPU states isolation**
  - CPU states at run-time are intrinsically isolated from other CPUs
- ➢ **Memory isolation**
  - PMP mechanisms isolate memory of different modes
- **I/O isolation**
  - IOPMP

Multiple C-mode ID

- **For global trusted secure shim model**
  - The seshim is the TCB that can isolate CVMs with G-stage page tables
    - Binary modes are sufficient to isolate seshim and HS-mode hypervisor

- **For per-CVM secure shim model**
  - Each CVM (with its seshim) can reside in an isolated mode
  - ➢ Multiple C-mode ID: each CVM is tagged with a unique ID

Design Direction

- **Previous (binary modes)**

63 60	59 56	55 52	51 48	47 44	43 40	39 36	35 32	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
perm15	perm14	perm13	perm12	perm11	perm10	perm9	perm8	perm7	perm6	perm5	perm4	perm3	perm2	perm1	perm0
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Format of leaf PMP table entry (RV64)

3	2	1	0
C (WARL)	X (WARL)	W (WARL)	R (WARL)
1	1	1	1

Permissions for a 4KiB physical page in leaf pmpte

- **Multiple C-mode ID**
  - Indicate C-mode ID in leaf PMP table entries

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
perm7	perm6	perm5	perm4	perm3	perm2	perm1	perm0

Format of leaf PMP table entry (RV64)

7 3	2	1	0
ID (WARL)	X (WARL)	W (WARL)	R (WARL)

Permissions for a 4KiB physical page in leaf pmpte

INTERACTION WITH AP-TEE

Interaction with AP-TEE

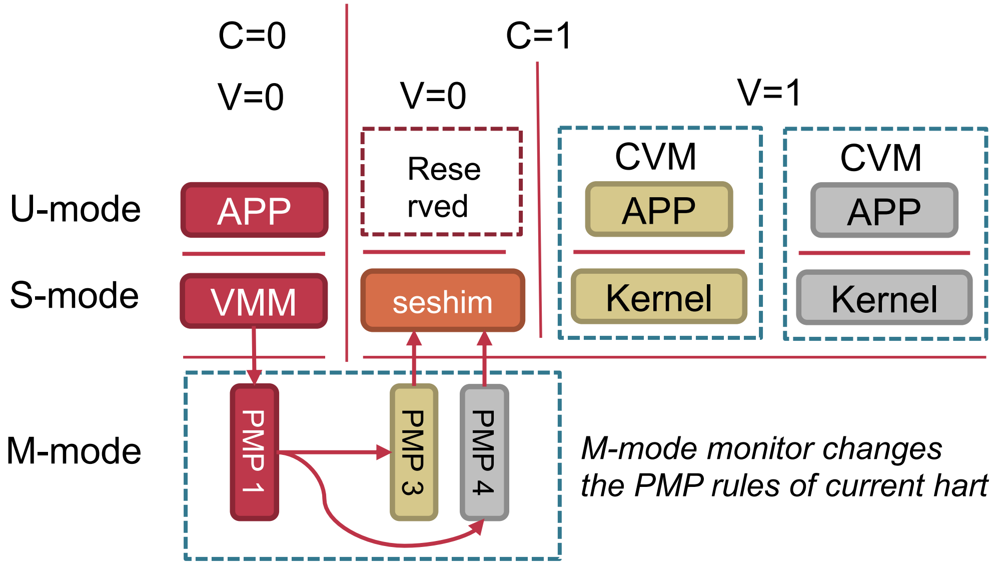
- **AP-TEE: a software guideline/arch based on existing hardware**
  - Switching between host OS/VMM and trusted OS/VMM via M-mode
  - ABIs: S-mode to M-mode, V-mode to HS-mode

- **Hardware extension is still necessary for confidential VM**
  - Attestation (local & remote)
  - Memory confidentiality and integrity
  - Fine-grained dynamic confidential memory adjustment
  - Performance improvement

PMP TABLE EXTENSION

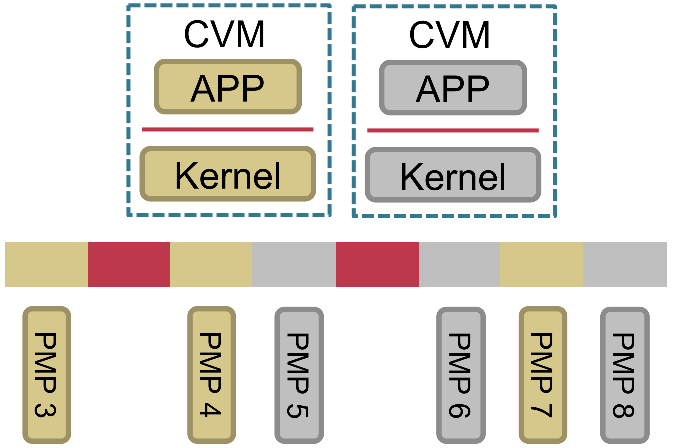
Problem Recap

- **Why not reuse existing PMP by changing PMP rule in M-mode?**
  - **Pros**: less modifications to existing spec if there are enough PMP entries
  - **Cons**: cannot support fine-grained memory accessibility



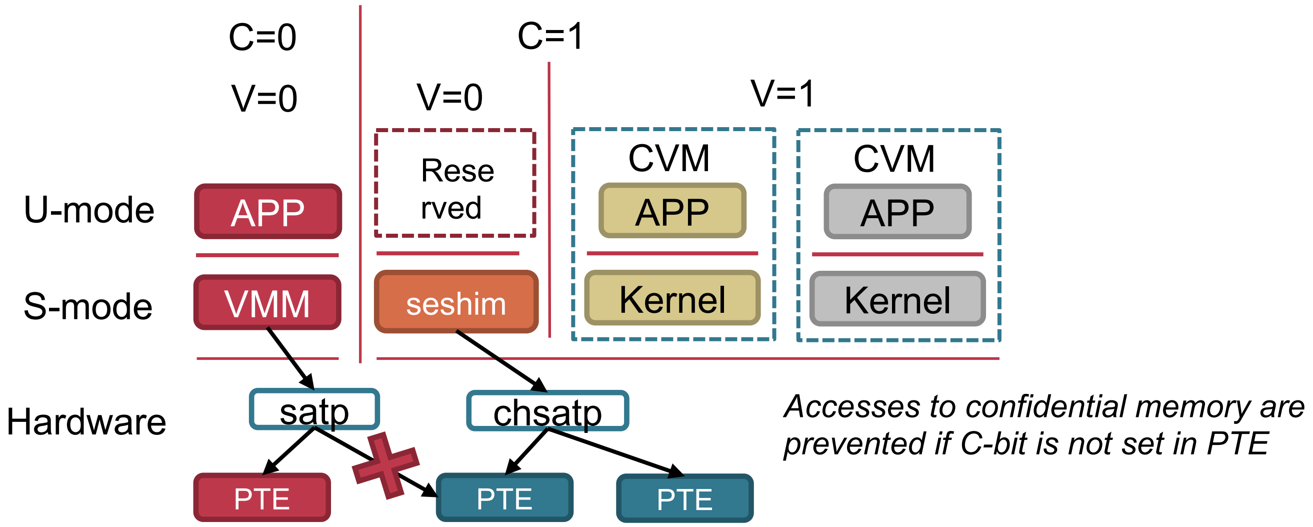
Existing PMP is Not Ideal

- **High resource utilization is one main goal of VMs**
  - Existing PMP entries: static memory partition / contiguous memory allocation
  - Memory demand of VMs: dynamic and nonconsecutive memory allocation
- **Limited number of PMP entries is inadequate**
  - For high memory utilization / high VM density
  - For fine-grained memory accessibility (e.g., 4KB granularity)



Problem from Google Doc

- **Why not reuse existing MMU by adding a C-bit in PTEs?**
  - **Pros**: small modifications to existing hardware, reuse cache and TLB
  - **Cons**: complicated software implementation



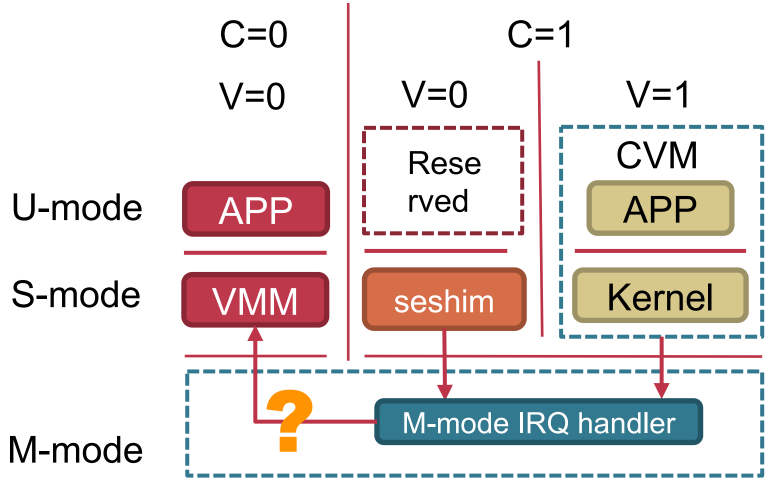
Trade-offs of Extending MMU

- **Must ensure the correctness of C-bit in PTEs**
  - Untrusted software can corrupt PTEs
- **Trusted components must control all modifications to PTEs**
  - Modify untrusted software, forward PT ops to trusted components
    - Pros: less modifications to hardware
    - Cons: intrusive modifications to untrusted software
  - Trap and emulate PT ops of untrusted software
    - Pros: transparent to untrusted software
    - Cons: excessive mode switches can result in poor performance
- **What if MMU is disabled by untrusted software?**

M-MODE INTERRUPT

Problem Recap

- **Can CHS-mode CSRs really reduce M-mode software complexity?**
  - Does M-mode still need to swap context when an M-mode interrupt happens?

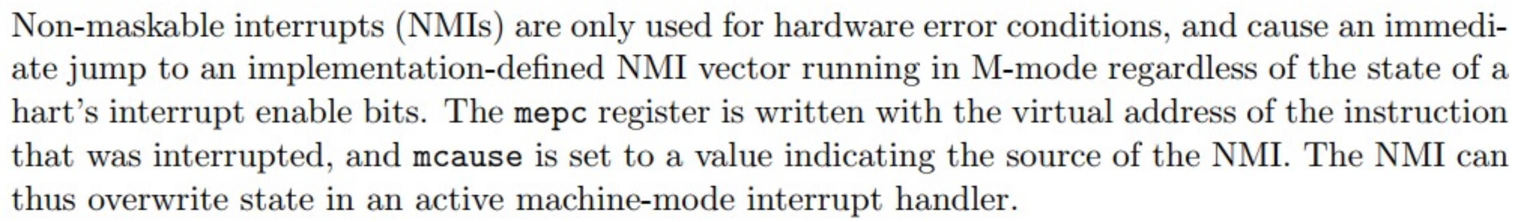


M-mode needs to save a copy of GP-registers in C mode, then enters non-C mode?

M-mode Interrupt Handling

- **M-mode interrupts**
  - Cause an immediate jump to the trap vector running in M-mode
  - Depend on the *mideleg, mie* and *mip*

- **Non-Maskable Interrupts (NMIs)**



Non-maskable interrupts (NMIs) are only used for hardware error conditions, and cause an immediate jump to an implementation-defined NMI vector running in M-mode regardless of the state of a hart's interrupt enable bits. The `mepc` register is written with the virtual address of the instruction that was interrupted, and `mcause` is set to a value indicating the source of the NMI. The NMI can thus overwrite state in an active machine-mode interrupt handler.

✓ **No additional context swap logic is needed**

1. Trap to M-mode and save GP-regs
2. Handle the interrupt in M-mode
3. Return to the trap point

Thanks!