OoH: Out of Hypervisor

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE NORMALE SUPÉRIEURE DE LYON

July 2nd, 2021

Ph.D. Student: Stella Bitchebe

Advisor: Pr. Alain Tchana

Inria

NEC

LIP Laboratory, ENS Lyon

1 Context

Intel PML 2

3 Intel SPP

OoH Implemention 4

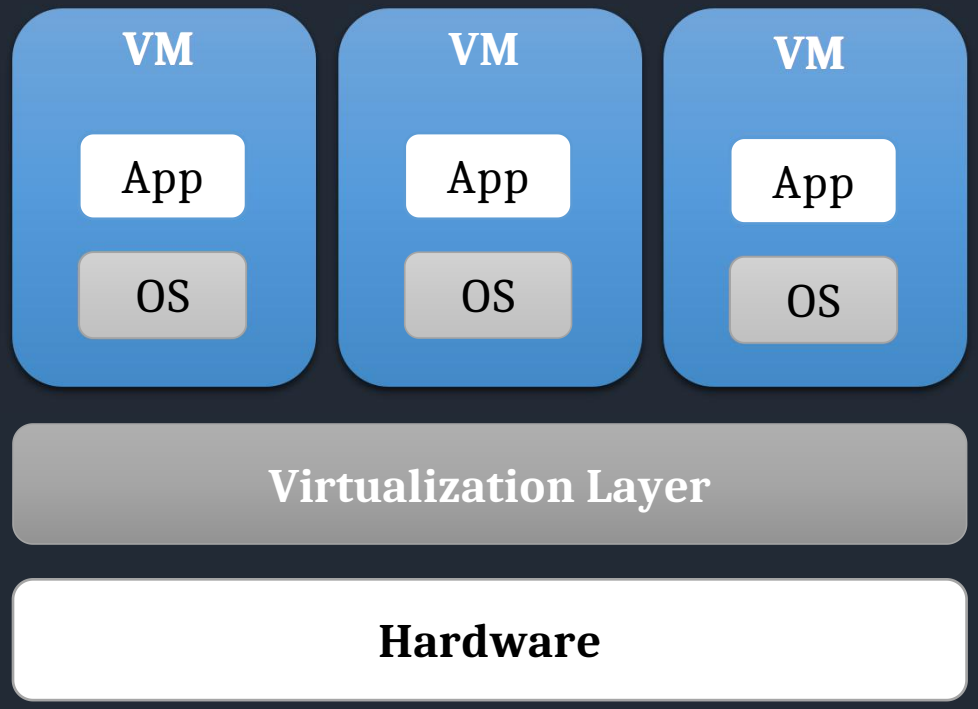
5 Conclusion

Context : Virtualization

Generalities

VM = Virtual Machine

Virtualization layer = Hypervisor



Virtualization Architecture

Context : Virtualization

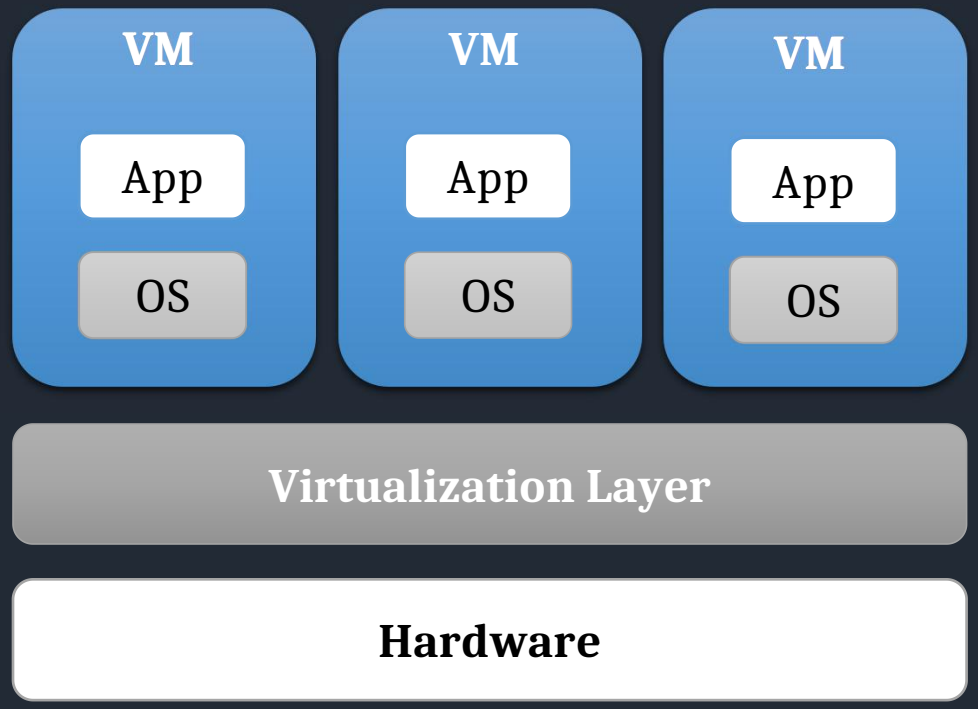
Generalities

VM = Virtual Machine

Virtualization layer = Hypervisor

Hypervisor role:

- Scheduling of VMs
- Memory Allocation
- Interrupt management
- etc.



Virtualization Architecture

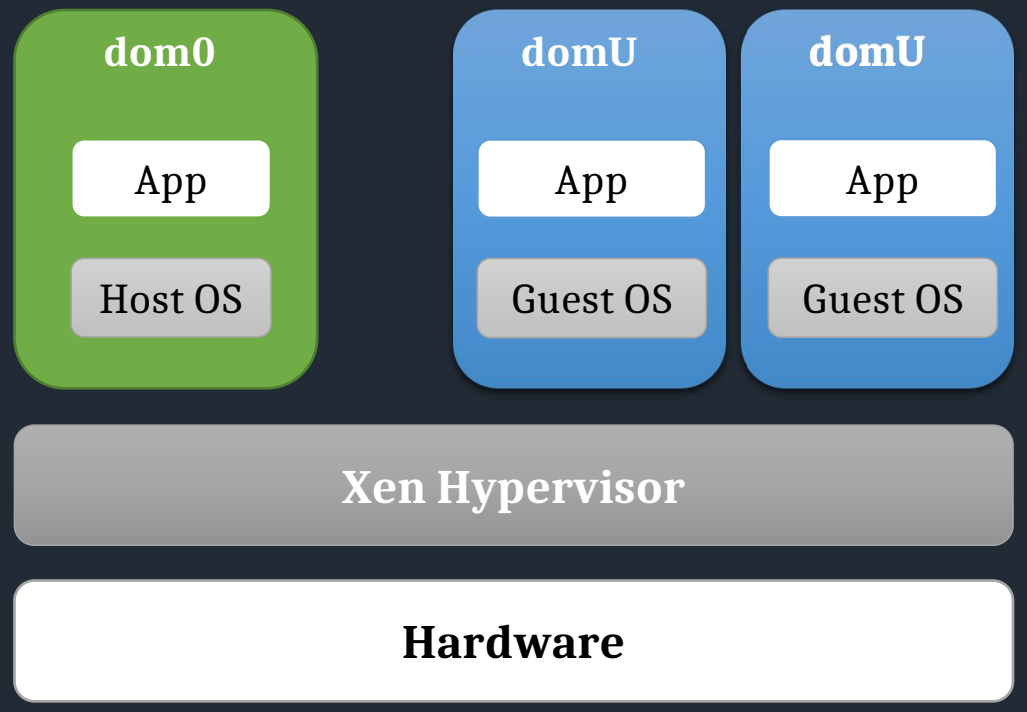
Context : Virtualization

Xen hypervisor

dom0 = Privileged domain

domU = Unprivileged domain

dom0 + hypv = Root context



Xen Architecture

Context : Virtualization

Xen hypervisor

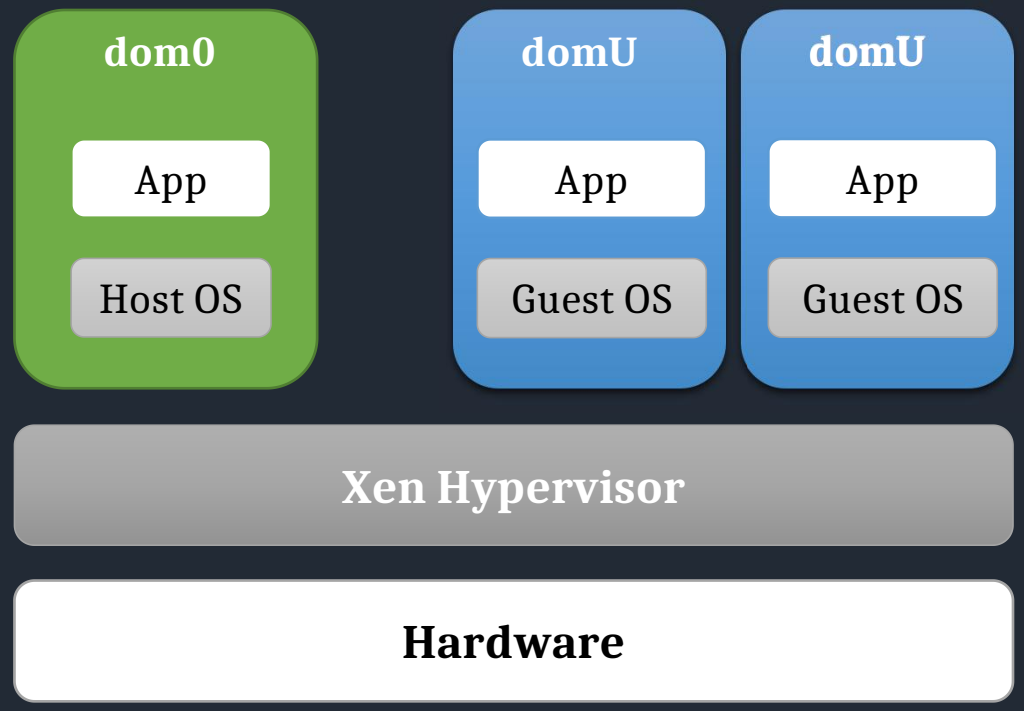
dom0 = Privileged domain

domU = Unprivileged domain

dom0 + hypv = Root context

Root context role:

- VM management tasks (creation, destruction, etc.)
- I/O drivers management
- Tools management (Xenstore -information storage space-, network, etc.)
- etc.



Xen Architecture

Context : Hardware Assisted Virtualization

Intel VT: Virtual Technology

- VT-x (Intel VT extension) - for **CPU virtualization**
- EPT (Extended Page Table) - for **Memory virtualization**
- VT-d (Intel VT for **direct I/O virtualization**)
- **etc.**

Context : Hardware Assisted Virtualization

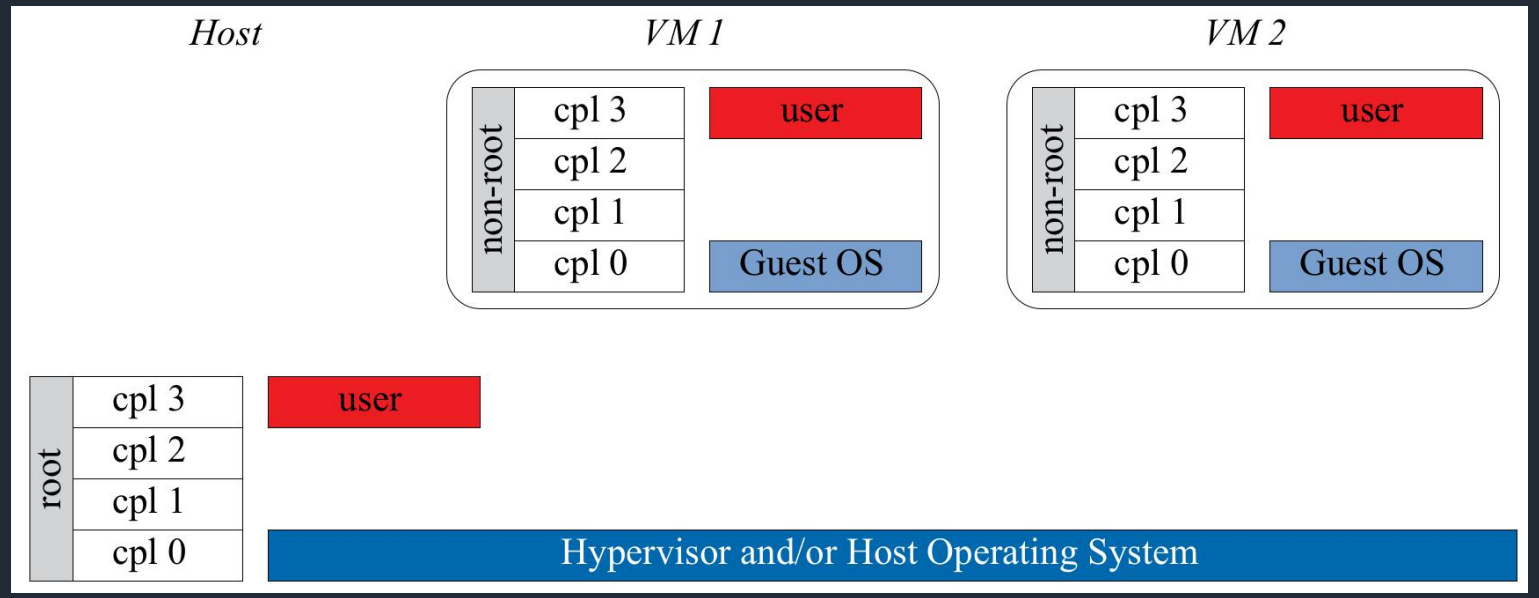
VT-x

- Provides architectural support for virtualization
- Based on a central design decision: **do not change** the semantics of individual instructions of the ISA (Instruction Set Architecture)
- Duplicates the entire architecturally visible state of the processor and introduces a new mode of execution: the **root mode**

Context : Hardware Assisted Virtualization

VT-x

- Provides architectural support for virtualization
- Based on a central design decision: **do not change** the semantics of individual instructions of the ISA (Instruction Set Architecture)
- Duplicates the entire architecturally visible state of the processor and introduces a new mode of execution: the **root mode**



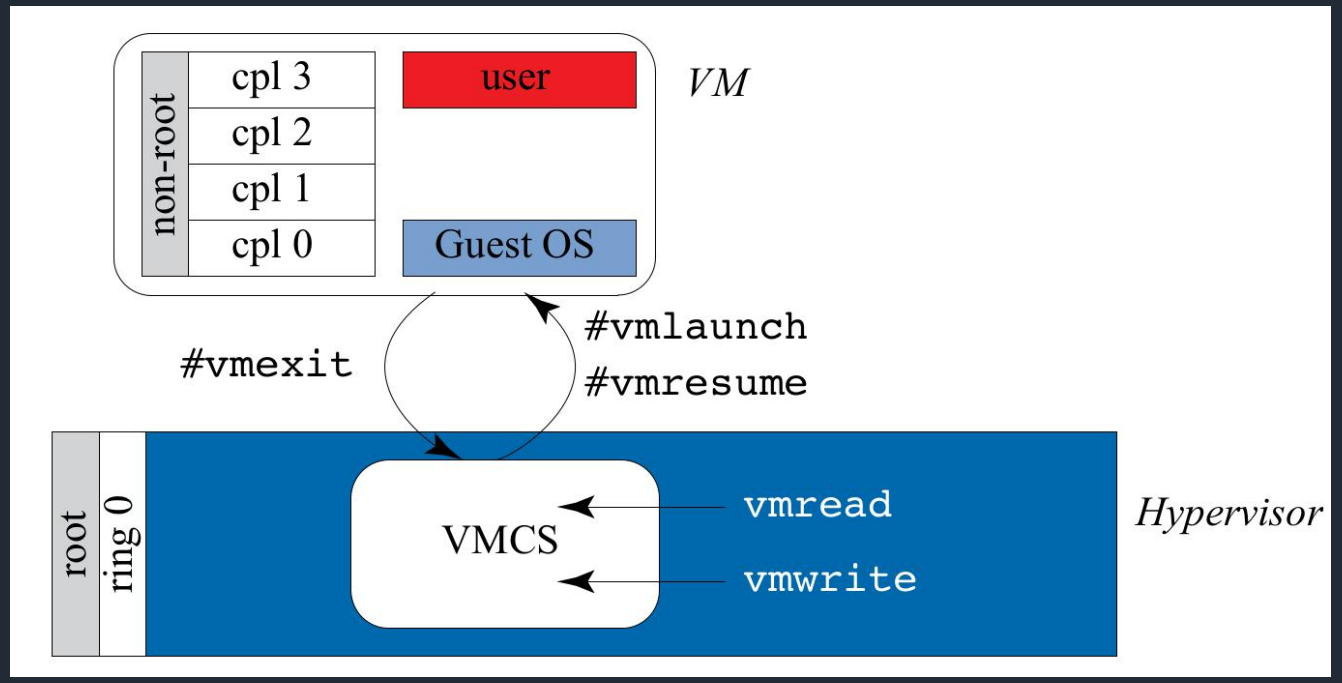
Standard use by hypervisors of VT-x root and non-root modes

(Src.: *Hardware and Software Support for Virtualization. Edouard Bugnion, Jason Nieh, Dan Tsafir*)

Context : Hardware Assisted Virtualization

VT-x Transitions

- vmexit: transition from non-root to root mode
- vmlaunch, vmresume: transition from root to non-root mode
- vmread, vmwrite: instruction to access the VMCS



VT-x transitions and control structures
(Src.: *Hardware and Software Support for Virtualization. Edouard Bugnion, Jason Nieh, Dan Tsafir*)

Context : Hardware Assisted Virtualization

VMCS: Virtual Machine Control Structure

In memory control structure that manages:

- Transitions into and out of VT-x operations (VM entries and VM exits)
- Processor behavior in VT-x non-root mode

Context : Hardware Assisted Virtualization

VMCS: Virtual Machine Control Structure

In memory control structure that manages:

- Transitions into and out of VT-x operations (VM entries and VM exits)
- Processor behavior in VT-x non-root mode

Manipulated by instructions (only in root mode):

- vmclear: clear aVMCS
- vmptrld: load a new VMCS
- vmread/vmwrite: read and write from and to a VMCS

Context : Hardware Assisted Virtualization

VMCS: Virtual Machine Control Structure

In memory control structure that manages:

- Transitions into and out of VT-x operations (VM entries and VM exits)
- Processor behavior in VT-x non-root mode

Manipulated by instructions (only in root mode):

- vmclear: clear aVMCS
- vmptrld: load a new VMCS
- vmread/vmwrite: read and write from and to a VMCS

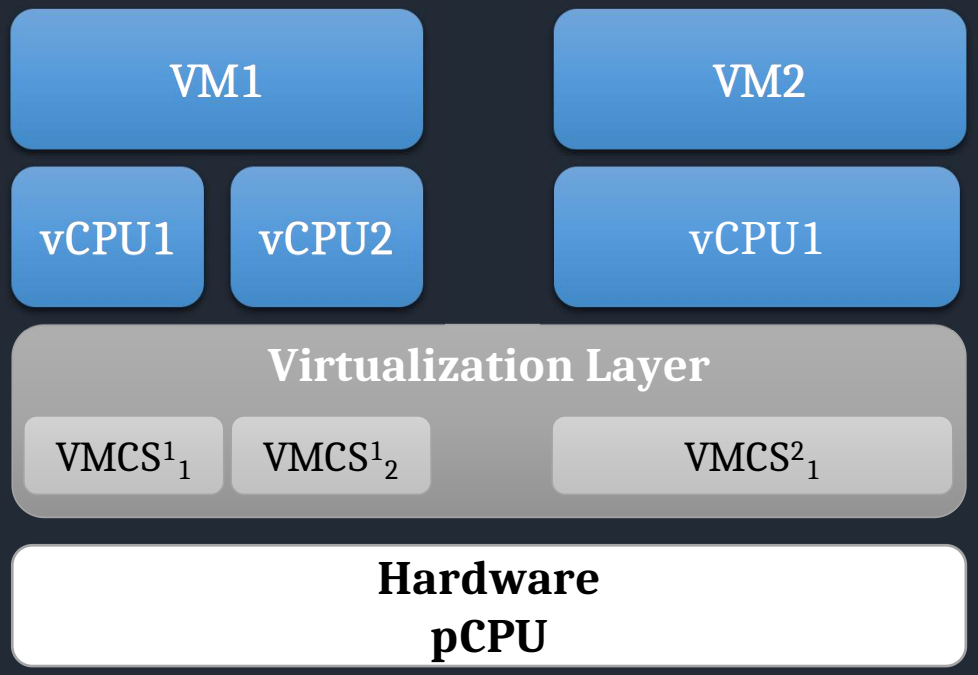
Can be of 2 types:

- Ordinary: normal VMCS used for vm entry, vmptrld, vmclear
- Shadow: can be accessed in non-root mode only by vmread and vmwrite instructions

Context : Hardware Assisted Virtualization

VMCS: Virtual Machine Control Structure

The hypervisor maintains a different VMCS per vCPU for each virtual machine



vCPU: virtual CPU

pCPU: physical/real CPU

Context : Hardware Assisted Virtualization

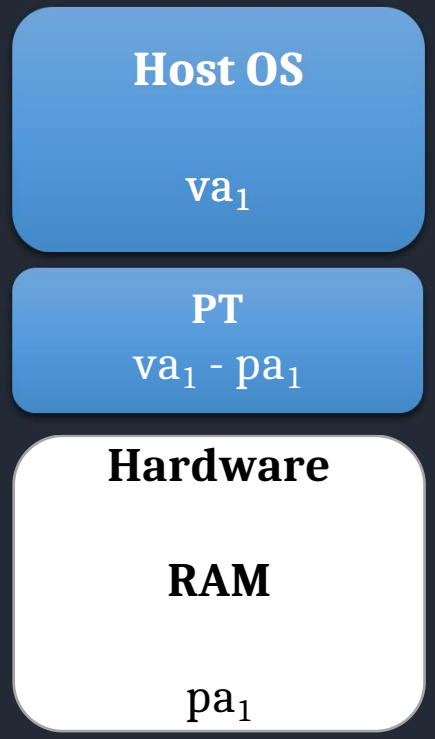
Intel VT: Virtual Technology

- VT-x (Intel VT extension) - for **CPU virtualization**
- EPT (Extended Page Table) - for **Memory virtualization**
- VT-d (Intel VT for **direct I/O virtualization**)
- **etc.**

Context : Hardware Assisted Virtualization

Memory Virtualization

- Bare Metal memory virtualization
  - va: virtual address
  - pa: physical address
  - PT: Page Table
    - va -> pa translation



Context : Hardware Assisted Virtualization

Memory Virtualization

- Bare Metal memory virtualization
  - va: virtual address
  - pa: physical address
  - PT: Page Table
    - va -> pa translation
- Virtualized environment memory virtualization
  - gva: guest virtual address
  - gpa: guest physical address
  - hpa: host physical address
  - gPT: guest Page Table
    - gva -> gpa translation

Context : Hardware Assisted Virtualization

Memory Virtualization: Shadow & Extended PTs

- Shadow Page Table
  - gva -> hpa translation

Context : Hardware Assisted Virtualization

Memory Virtualization: Shadow & Extended PTs

- Shadow Page Table (SPT)
  - gva -> hpa translation
- Extended Page Table (EPT)
  - gva -> gpa translation (guest page table)
  - gpa -> hpa translation (EPT)

Context : Hardware Assisted Virtualization

EPT-based Intel VT features

- EPT-based functionalities
  - Intel PML: Page Modification Logging (2015)
  - Intel Sub-Page write Permission (2018)

Intel PML : Page Modification Logging

Aim

- Page tracking:
  - Checkpointing for recovery after failure **-> saving the state of a virtual machine at a given time**
  - Live migration for maintainance **-> move a virtual machine from one physical node to another**
  - Working set size estimation for memory overcommitment **-> packing more virtual machines on a physical node**

Intel PML : Page Modification Logging

Aim

- Page tracking:
  - Checkpointing for recovery after failure **-> saving the state of a virtual machine at a given time**
  - Live migration for maintainance **-> move a virtual machine from one physical node to another**
  - Working set size estimation for memory overcommitment **-> packing more virtual machines on a physical node**

- Traditionnal technique:
  - Present-bit/Dirty-bit invalidation
  - Costly page fault handling

Intel PML : Page Modification Logging

Aim

- Page tracking:
  - Checkpointing for recovery after failure **-> saving the state of a virtual machine at a given time**
  - Live migration for maintainance **-> move a virtual machine from one physical node to another**
  - Working set size estimation for memory overcommitment **-> packing more virtual machines on a physical node**

- Traditionnal technique:
  - Present-bit/Dirty-bit invalidation
  - Costly page fault handling

- PML:
  - Page tracking technique without need for invalidating present bit of pages

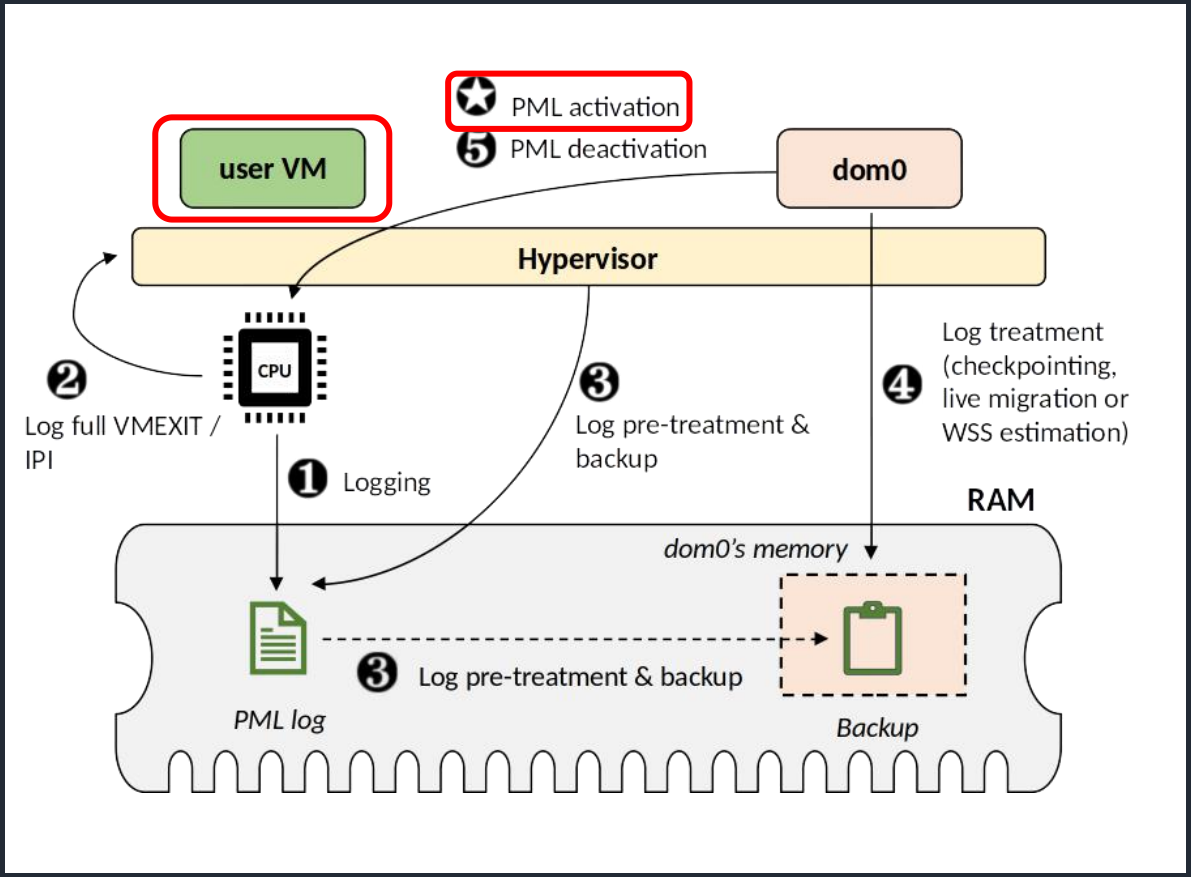
Intel PML : Page Modification Logging

Overview

- Allows the hypervisor to track write memory accesses of guests
- VMCS changes (introduction of new fields):
  - PML address
  - PML index
- VT-x transitions changes:
  - New vmexit reason

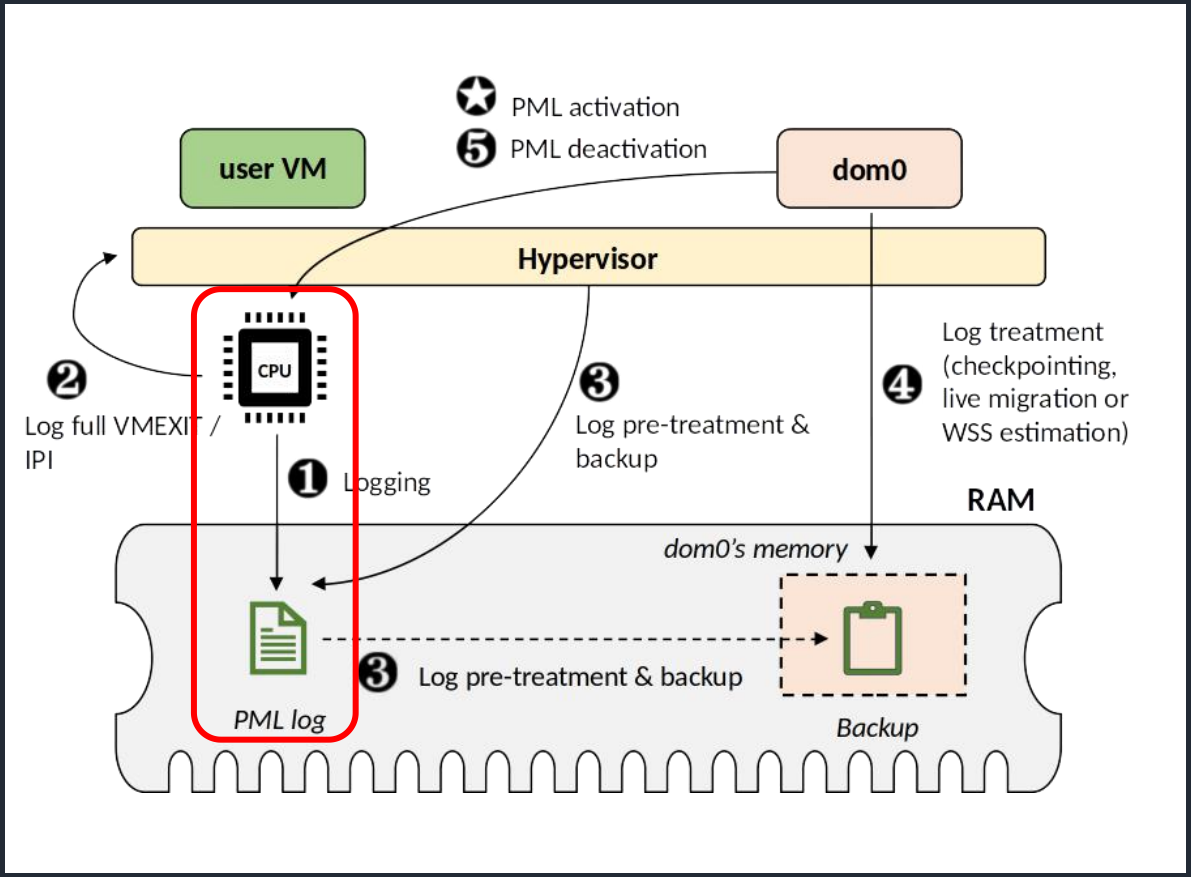
Intel PML : Page Modification Logging

Functioning



Intel PML : Page Modification Logging

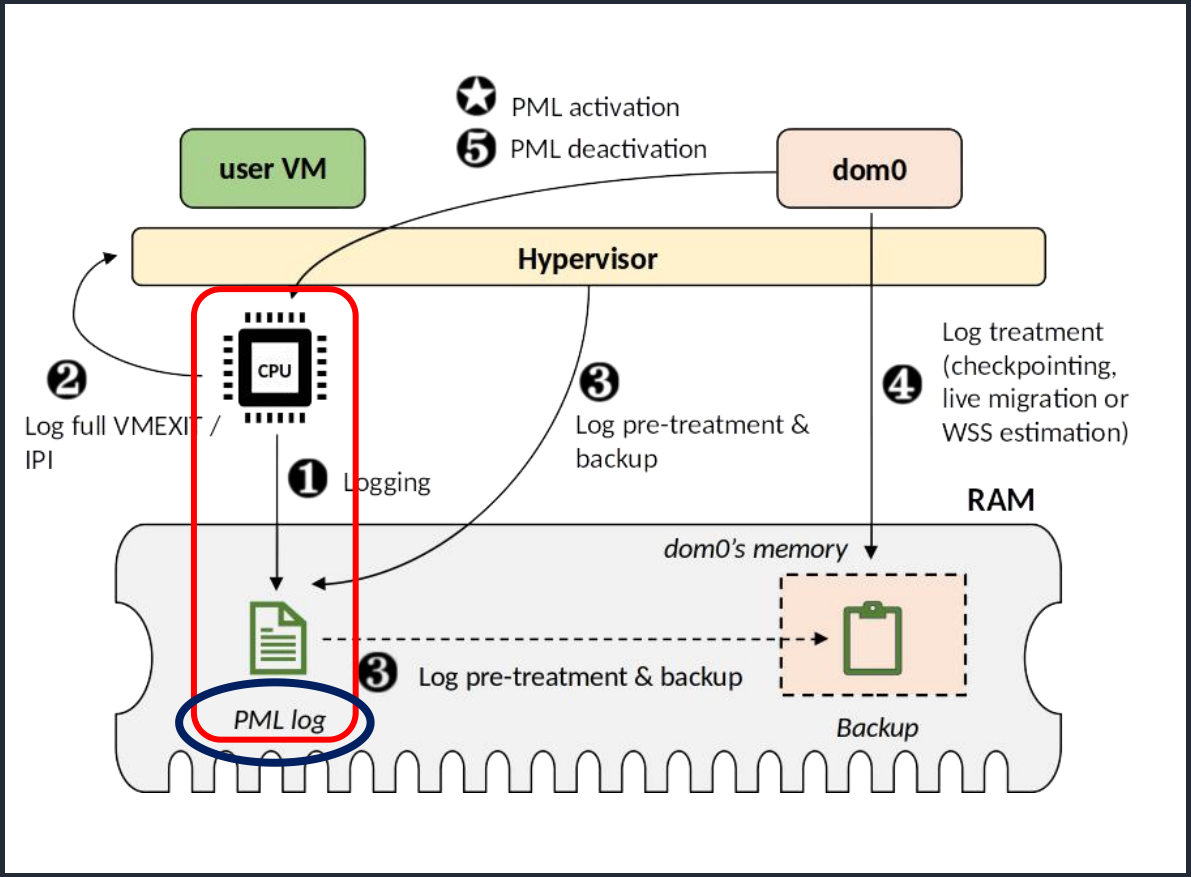
Functioning



Intel PML : Page Modification Logging

Functioning

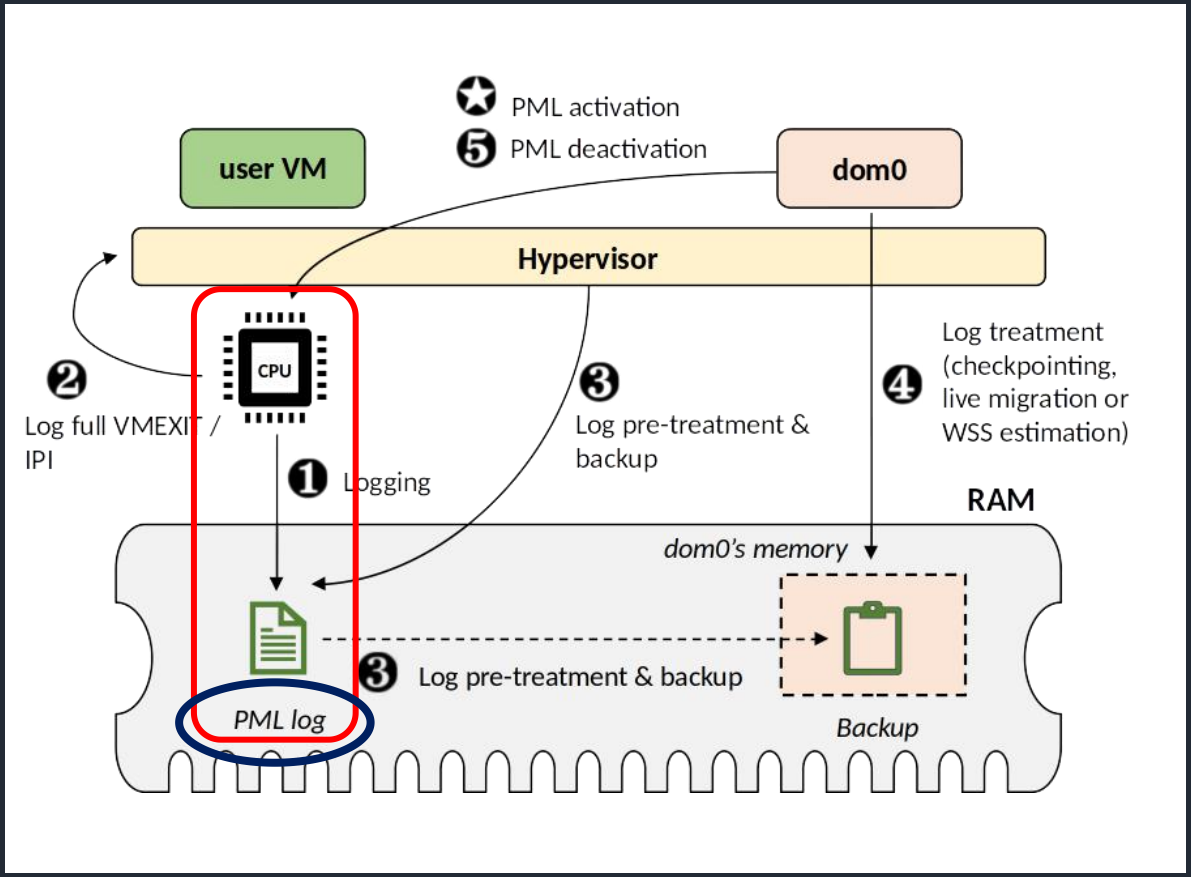
- PML log:
  - 4KB page in RAM
  - where the gpas are stored
- PML address:
  - address of the PML log stored in the VMCS (new field introduced for the PML mechanism)



Intel PML : Page Modification Logging

Functioning

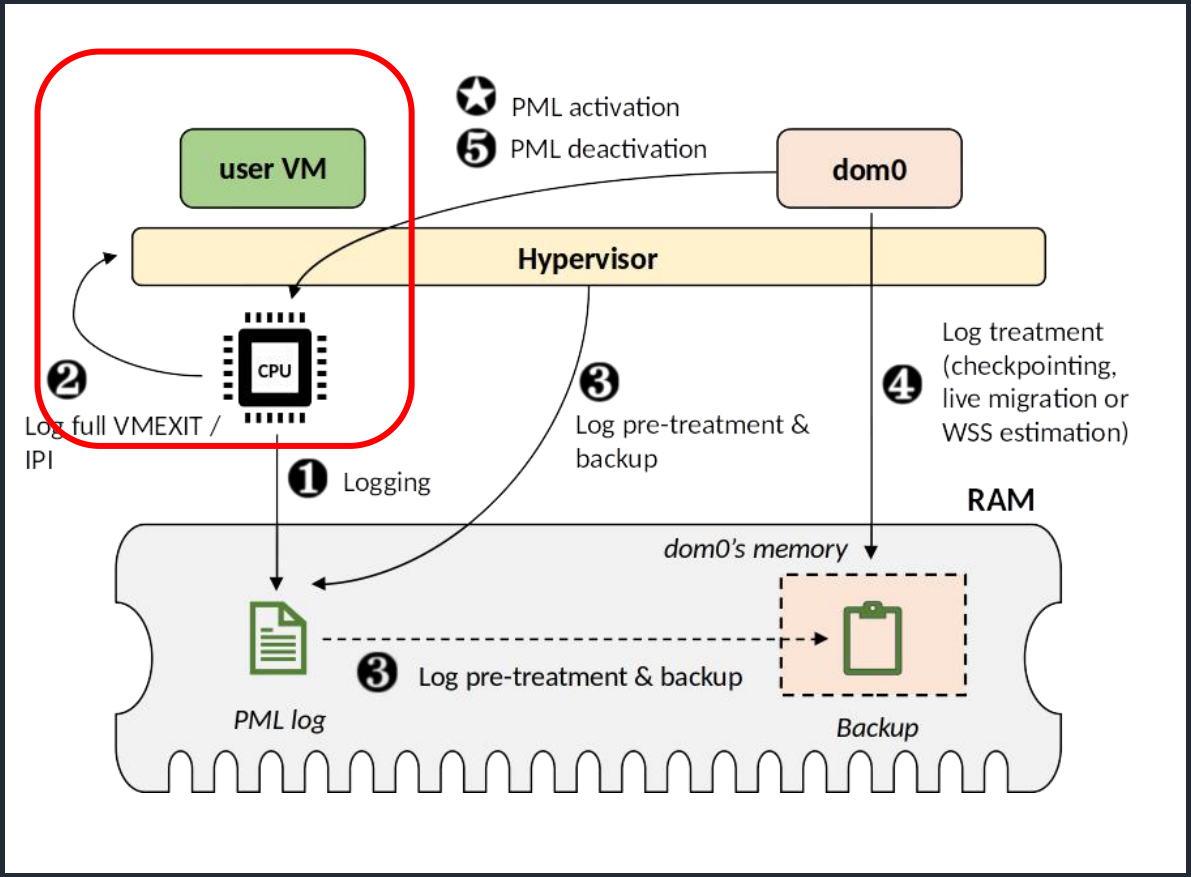
- PML log:
  - 4KB page in RAM
  - where the gpas are stored
- PML address:
  - address of the PML log stored in the VMCS
- PML index:
  - stored in the VMCS
  - value in range 0 - 511 (starting from 511)



Intel PML : Page Modification Logging

Functioning

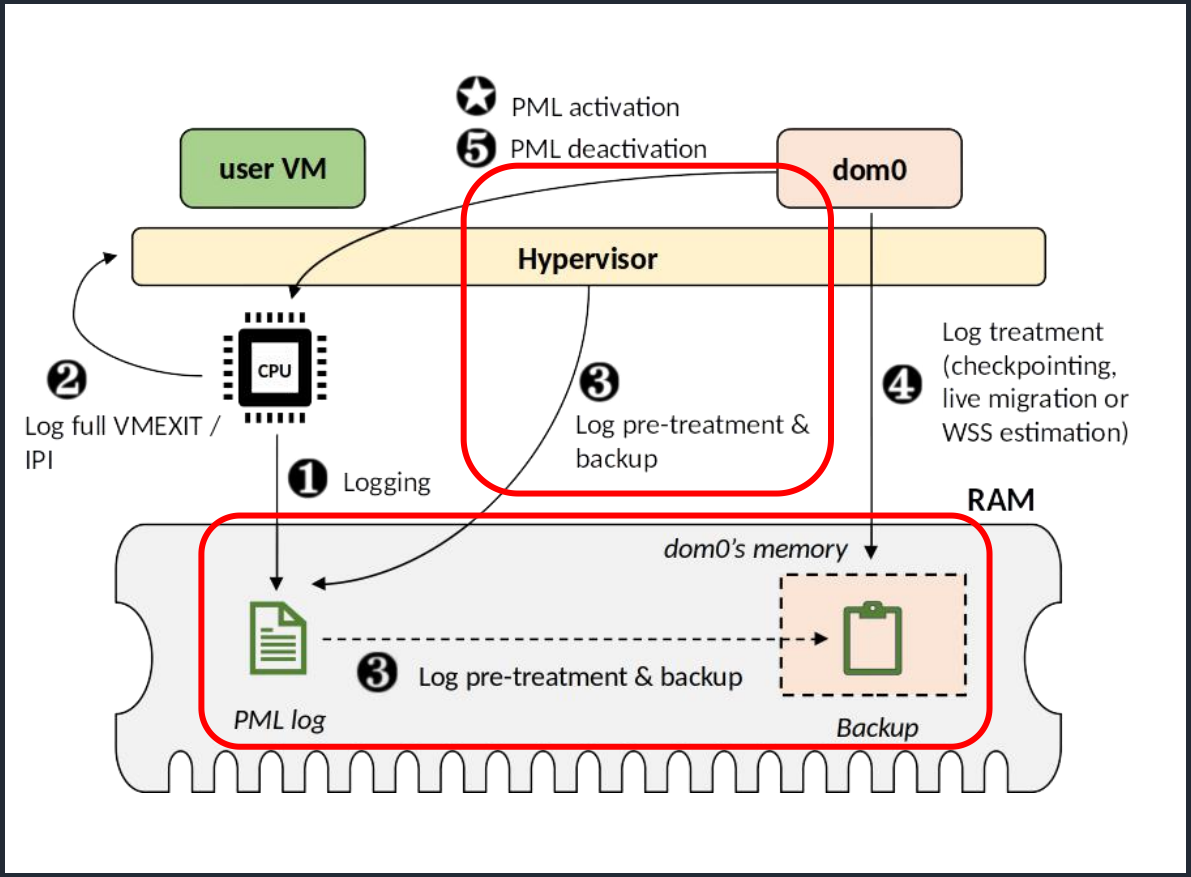
- PML log full:
  - vmexit reason introduced for PML mechanism
  - PML log is full after 512 addresses logged: PML index < 0



Intel PML : Page Modification Logging

Functioning

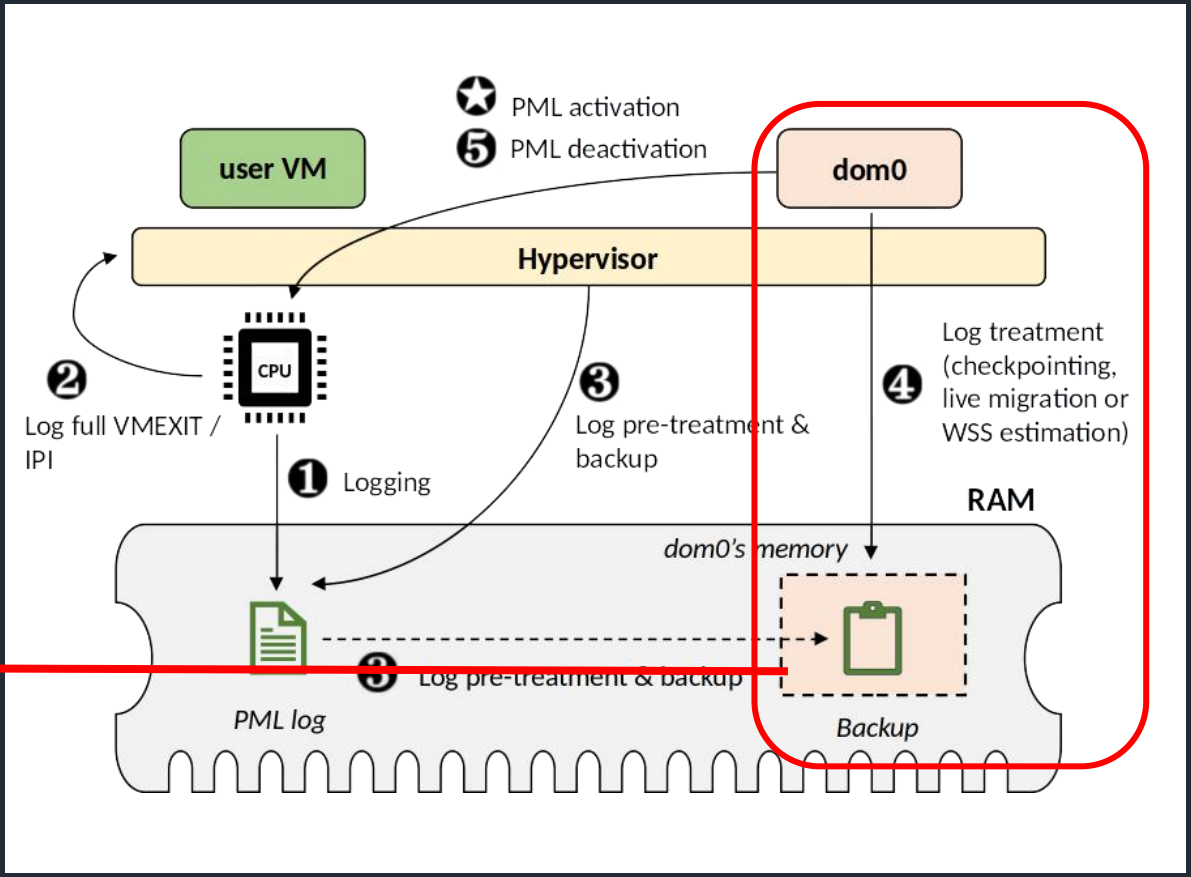
- PML log full vmexit, hypervisor:
  - flushes the log (copies the content to a larger buffer)
  - resets the PML index to 511
  - resumes VM execution (vmentry)



Intel PML : Page Modification Logging

Functioning

- Root-context uses the addresses to perform an operation
  - live migration
  - working set estimation

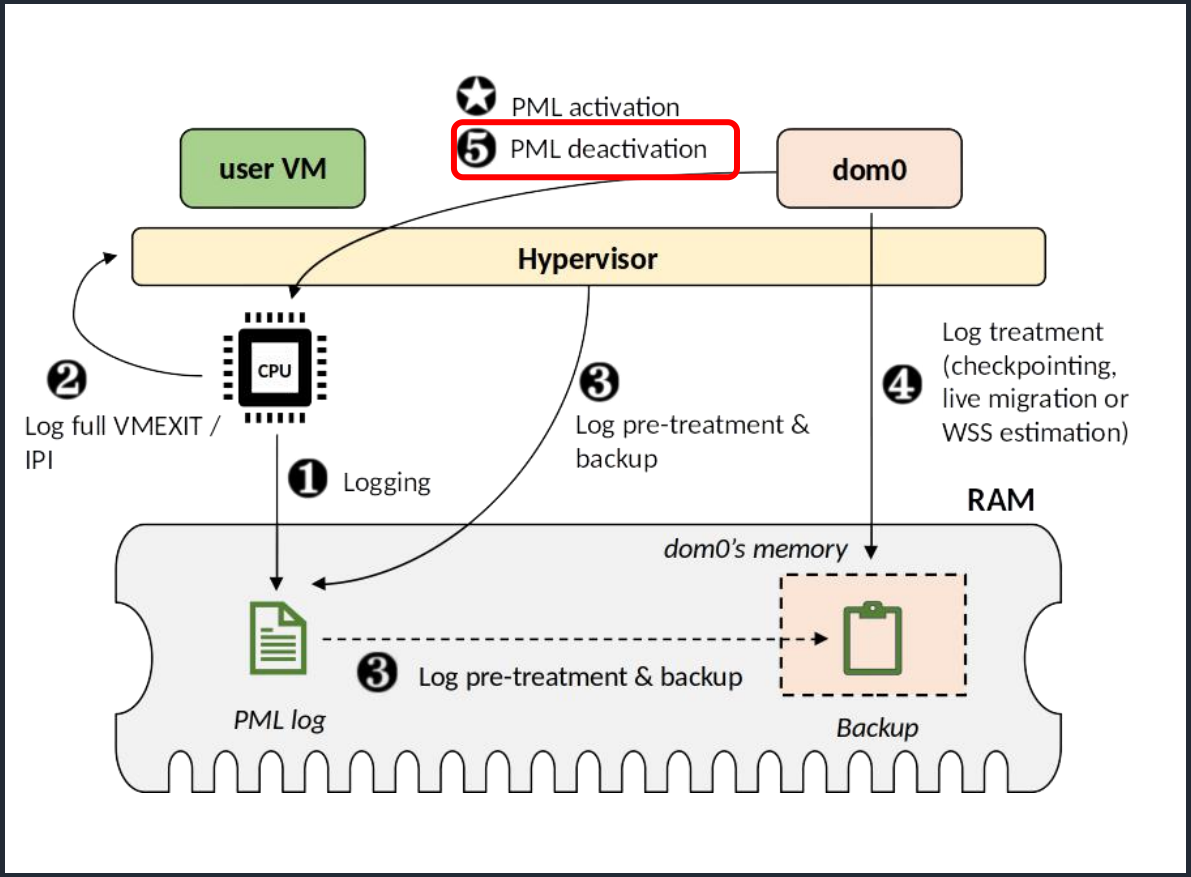


- gpa_1
- gpa_2
- ...
- ...
- gpa_n

Intel PML : Page Modification Logging

Functioning

- Deactivation:
  - disable PML for the VM
  - logging process stops
  - no more vmexit due to PML



Intel PML : Page Modification Logging

Usage

- Xen:
  - Implements PML for live migration of VMs

- Previous work [1], [2]:
  - VM's WSS estimation system based on PML

[1] Hardware Assisted Virtual Machine Page Tracking. Stella Bitchebe, Djob Mvondo, Alain Tchana, Laurent Réveillère, Noel De Palma. COMPAS'19
[2] Extending Intel PML for Hardware-Assisted Working Set Size Estimation of VMs. Stella Bitchebe, Djob Mvondo, Alain Tchana, Laurent Réveillère, Noel De Palma. VEE'21

Intel PML : Page Modification Logging

Usage

- Xen:
  - Implements PML for live migration of VMs
- Previous work [1], [2]:
  - VM's WSS estimation system based on PML
- For applications inside the VM:
  - Live migration of containers
  - WSS estimation of applications
  - Checkpoint/Restore of containers/applications

[1] Hardware Assisted Virtual Machine Page Tracking. Stella Bitchebe, Djob Mvondo, Alain Tchana, Laurent Réveillère, Noel De Palma. COMPAS'19
[2] Extending Intel PML for Hardware-Assisted Working Set Size Estimation of VMs. Stella Bitchebe, Djob Mvondo, Alain Tchana, Laurent Réveillère, Noel De Palma. VEE'21

Intel SPP: Sub-Page write Permission

Aim

*This part of the work is done with **Yves Kone**, a research engineer in our team*

Intel SPP: Sub-Page write Permission

Aim

- Page protection:
  - Buffer overflow mitigation

*This part of the work is done with **Yves Kone**, a research engineer in our team*

Intel SPP: Sub-Page write Permission

Aim

- Page protection:
  - Buffer overflow mitigation
- State of the art techniques:
  - Guard pages **-> 4KB**
- Problem:
  - Costly page fault handling
  - Memory waste

*This part of the work is done with **Yves Kone**, a research engineer in our team*

Intel SPP: Sub-Page write Permission

Aim

- Page protection:
  - Buffer overflow mitigation
- State of the art techniques:
  - Guard pages **-> 4KB**
- Problem:
  - Costly page fault handling
  - Memory waste
- SPP:
  - Protect a sub-page: 128 octets instead of 4KB (entire page)

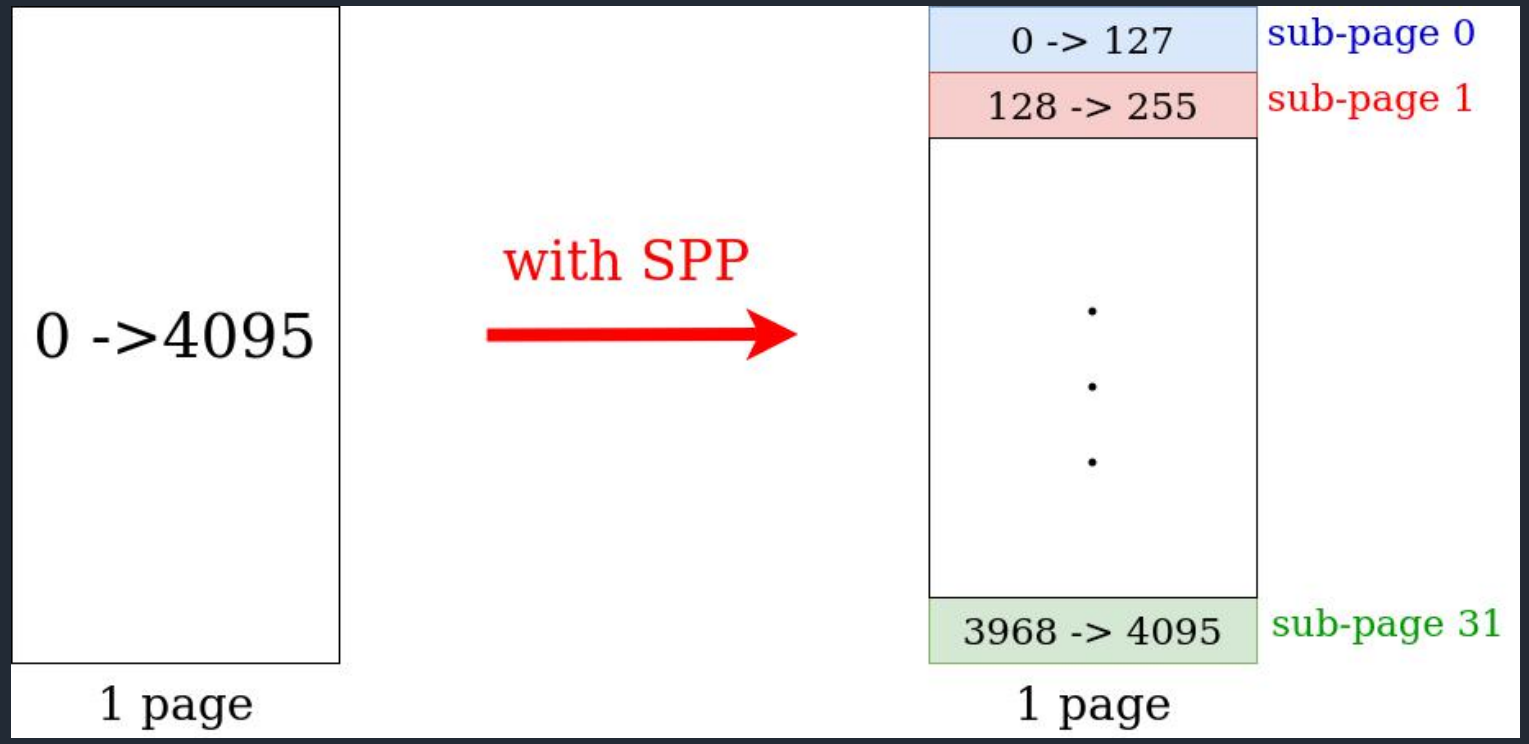
*This part of the work is done with **Yves Kone**, a research engineer in our team*

Intel SPP: Sub-Page write Permission

Overview

Allows the hypervisor to specify write-permission for guest physical memory at a sub-page(128 bytes) granularity

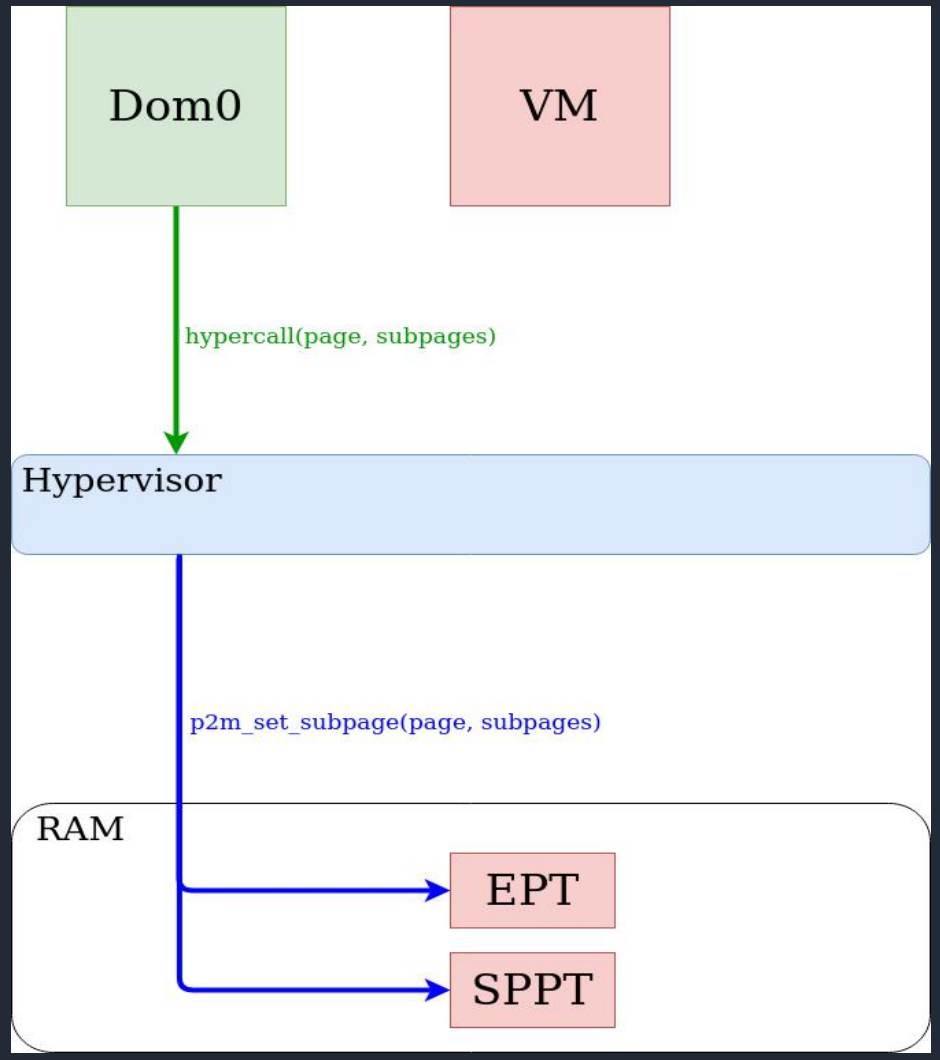
A new level page table walk : SPP Table



*This part of the work is done with **Yves Kone**, a research engineer in our team*

Intel SPP: Sub-Page write Permission

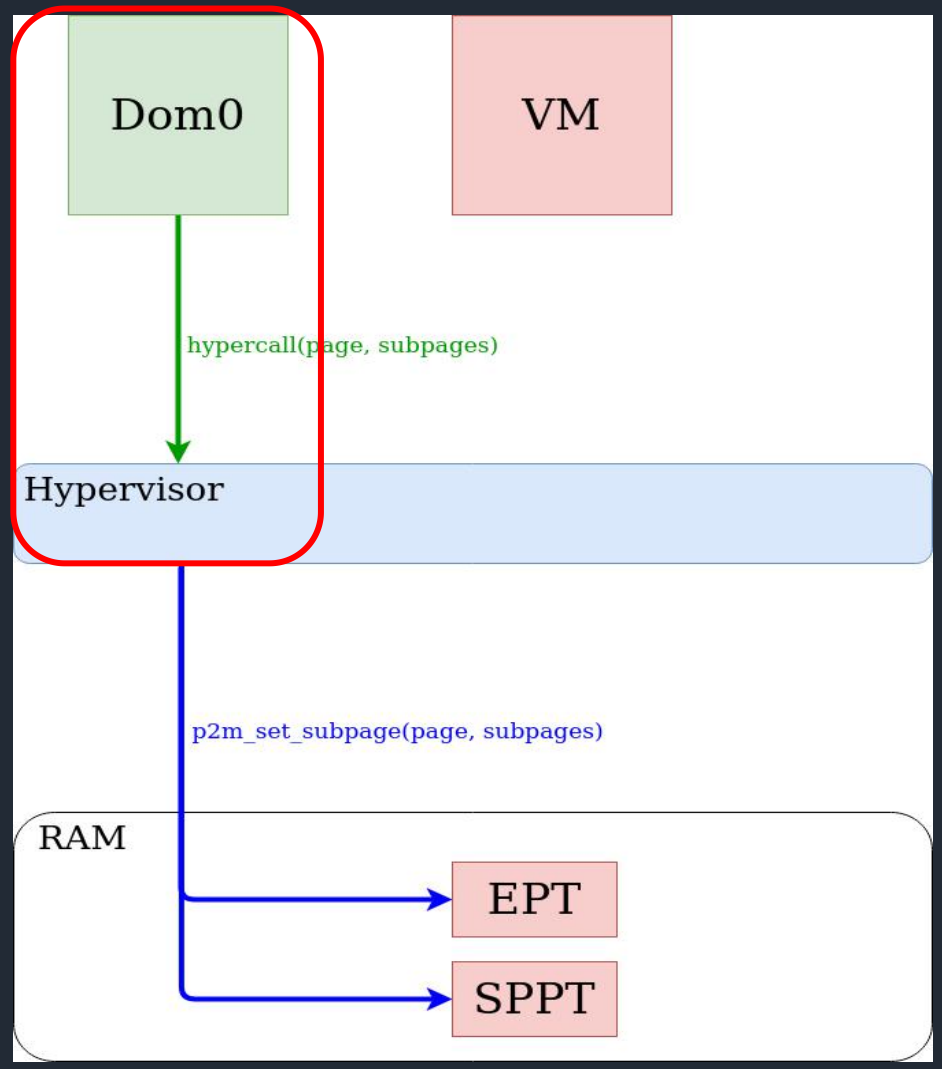
Functioning



*This part of the work is done with **Yves Kone**, a research engineer in our team*

Intel SPP: Sub-Page write Permission

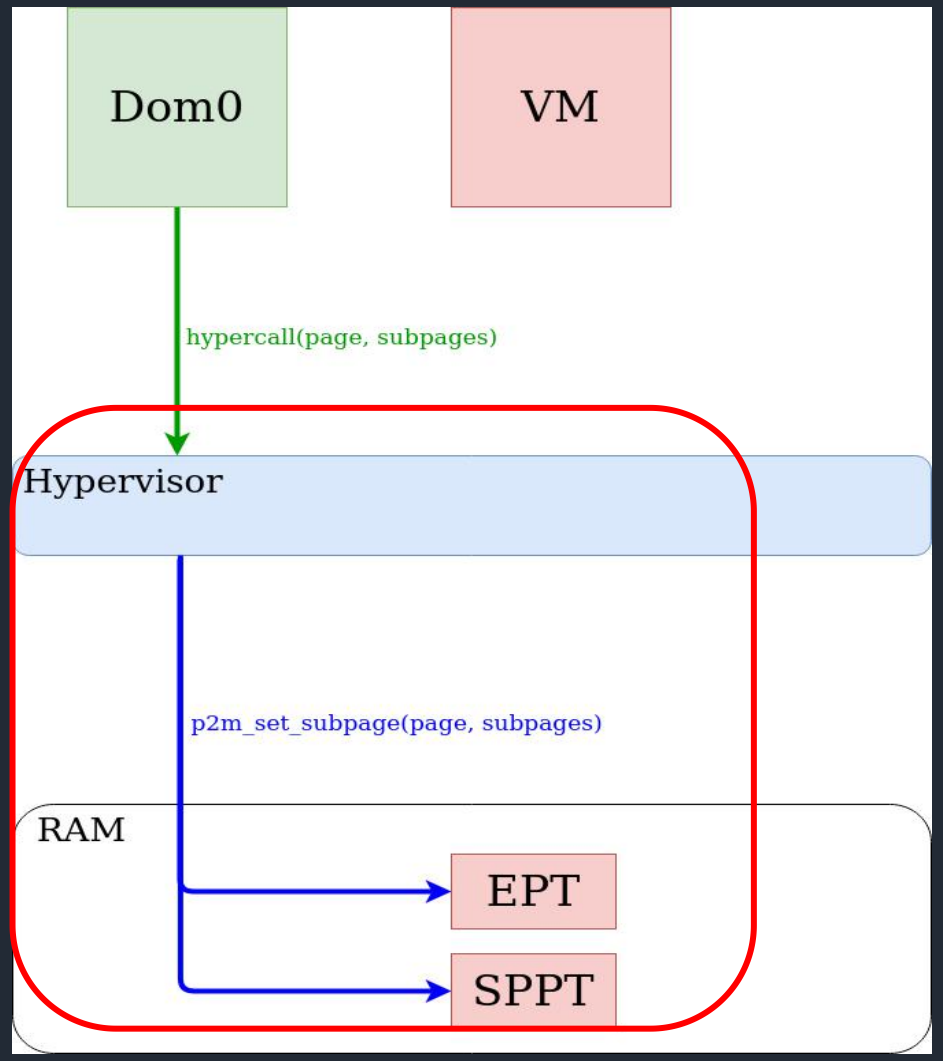
Functioning



*This part of the work is done with **Yves Kone**, a research engineer in our team*

Intel SPP: Sub-Page write Permission

Functioning



*This part of the work is done with **Yves Kone**, a research engineer in our team*

OoH: Out of Hypervisor

Overview

- Context:
    - Some hardware features may be useful in guest user space (inside the virtual machine)

OoH: Out of Hypervisor

Overview

- Context:
  - Some hardware features may be useful in guest user space (inside the virtual machine)
- Use cases (recall):
  - Checkpoint/Restore of containers
  - Memory protection (buffer overflow mitigation)
  - etc.

OoH: Out of Hypervisor

Overview

- Context:
  - Some hardware features may be useful in guest user space (inside the virtual machine)
- Use cases (recall):
  - Checkpoint/Restore of containers
  - Memory protection (buffer overflow mitigation)
  - etc.
- Problem:
  - Guest doesn't have direct access to the hardware

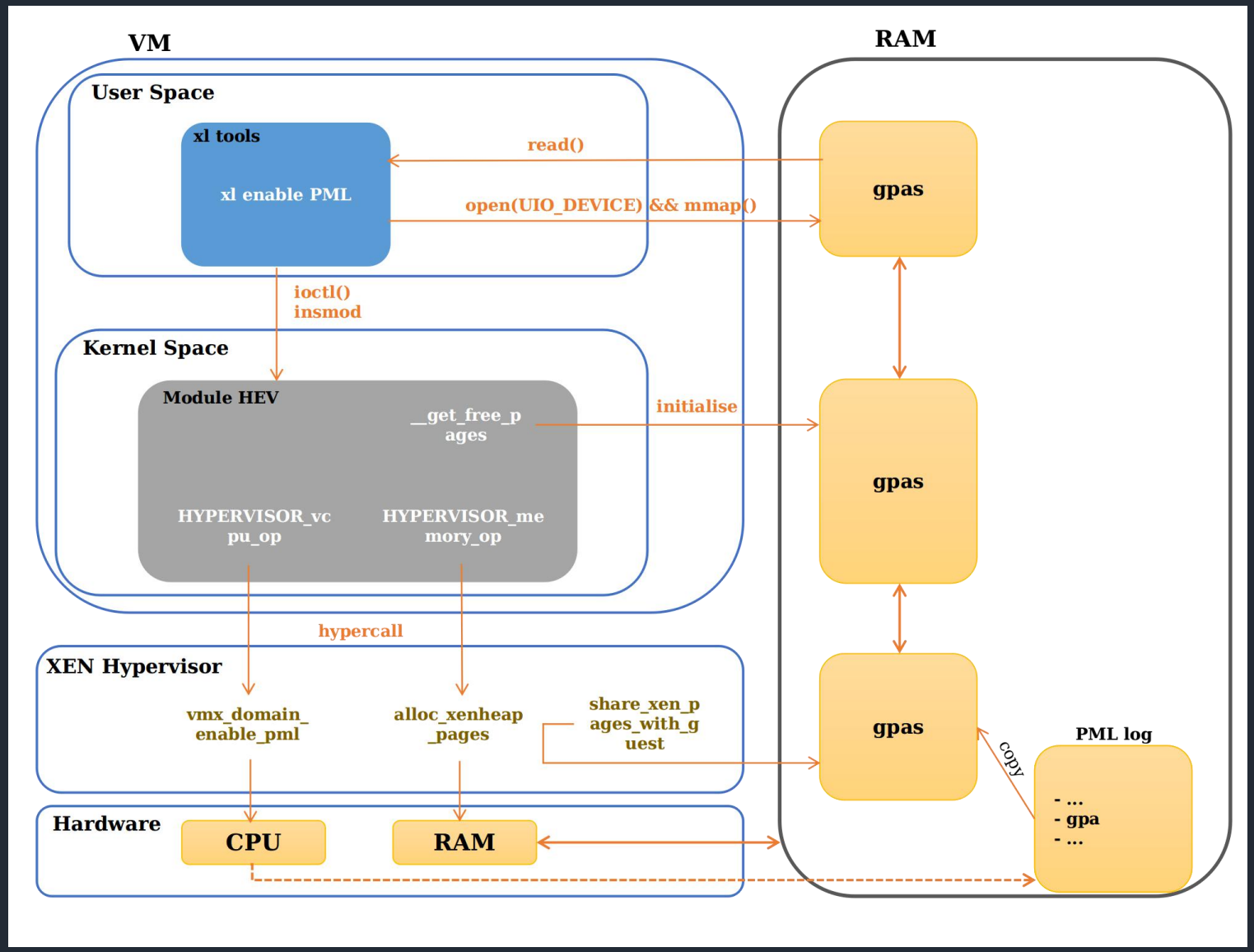
OoH: Out of Hypervisor

PML

- 4 implementations S0 -> S3
- A kernel module and an API
- Bochs Emulator

OoH: Out of Hypervisor

PML: S0 design



OoH: Out of Hypervisor

PML: S0 limits

- Hypervisor modification

OoH: Out of Hypervisor

PML: S0 limits

- Hypervisor modification
- Hypercalls

OoH: Out of Hypervisor

PML: S0 limits

- Hypervisor modification
- Hypercalls
- virq (virtual interrupt requests)

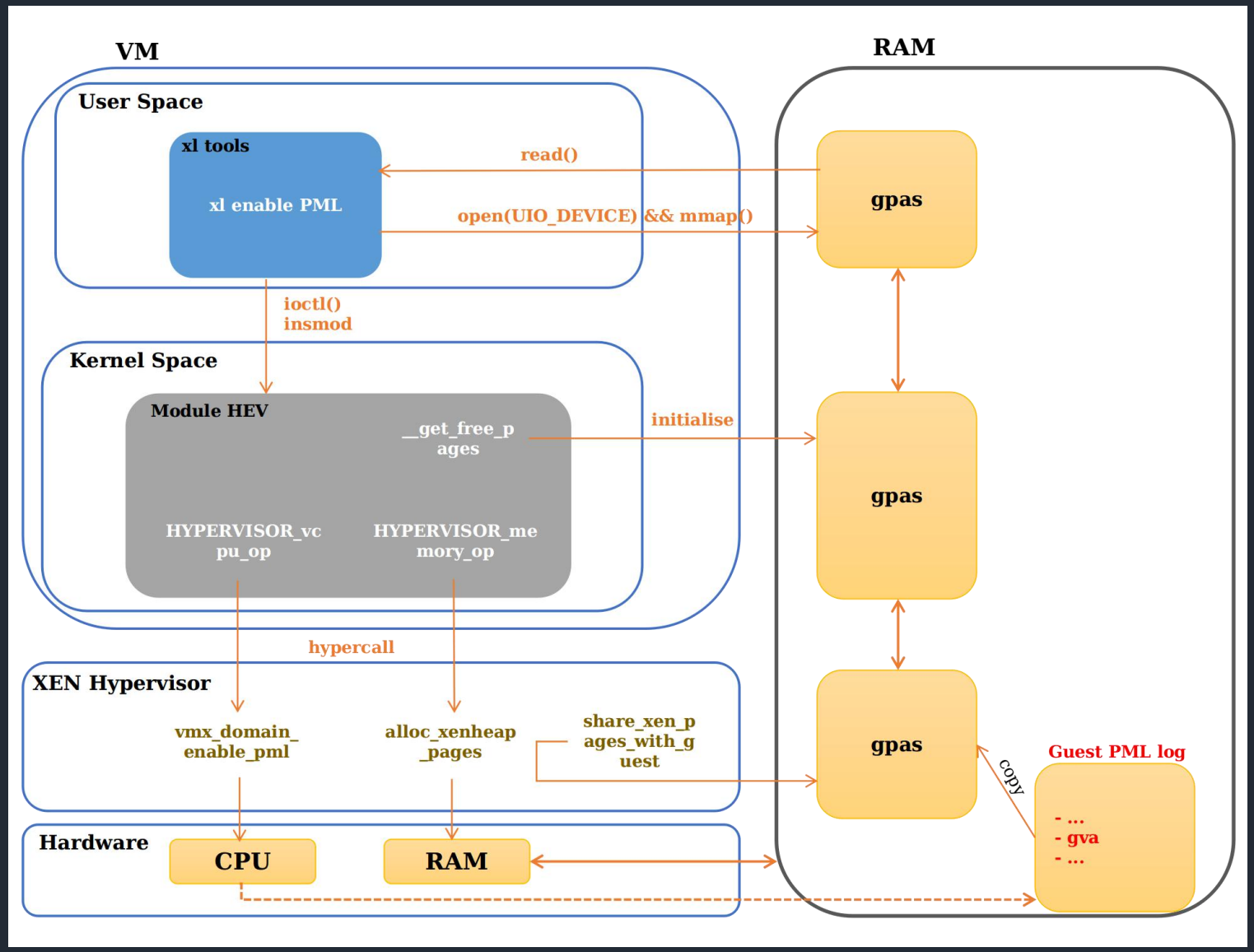
OoH: Out of Hypervisor

PML: S0 limits

- Hypervisor modification
- Hypercalls
- virq (virtual interrupt requests)
- Reverse mapping

OoH: Out of Hypervisor

PML: S1 design



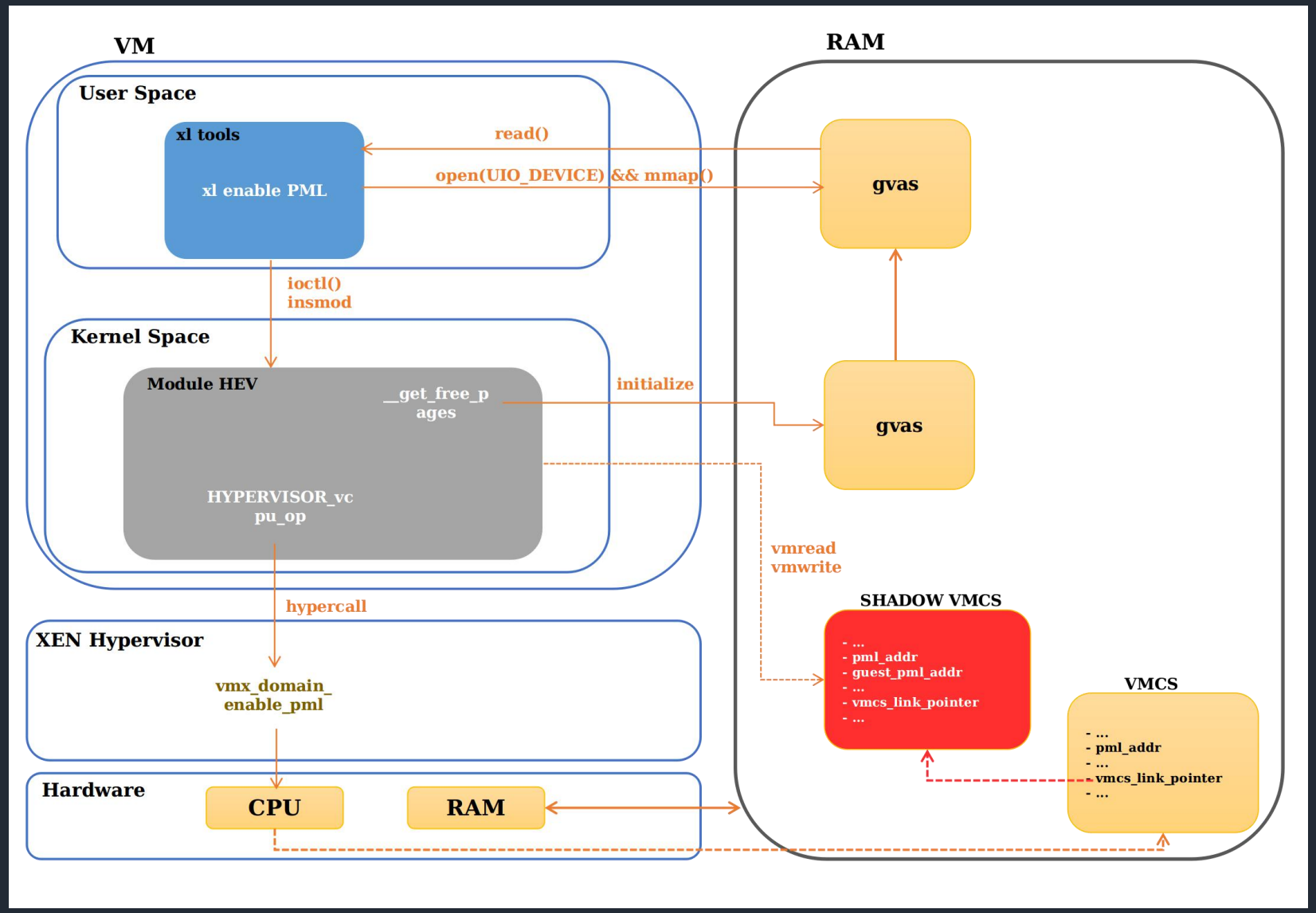
OoH: Out of Hypervisor

PML: S1 limits

- Hypervisor modification
- HW modification
- Hypercalls
- virq (virtual interrupt requests)
- Reverse mapping ✖

OoH: Out of Hypervisor

PML: S2 design



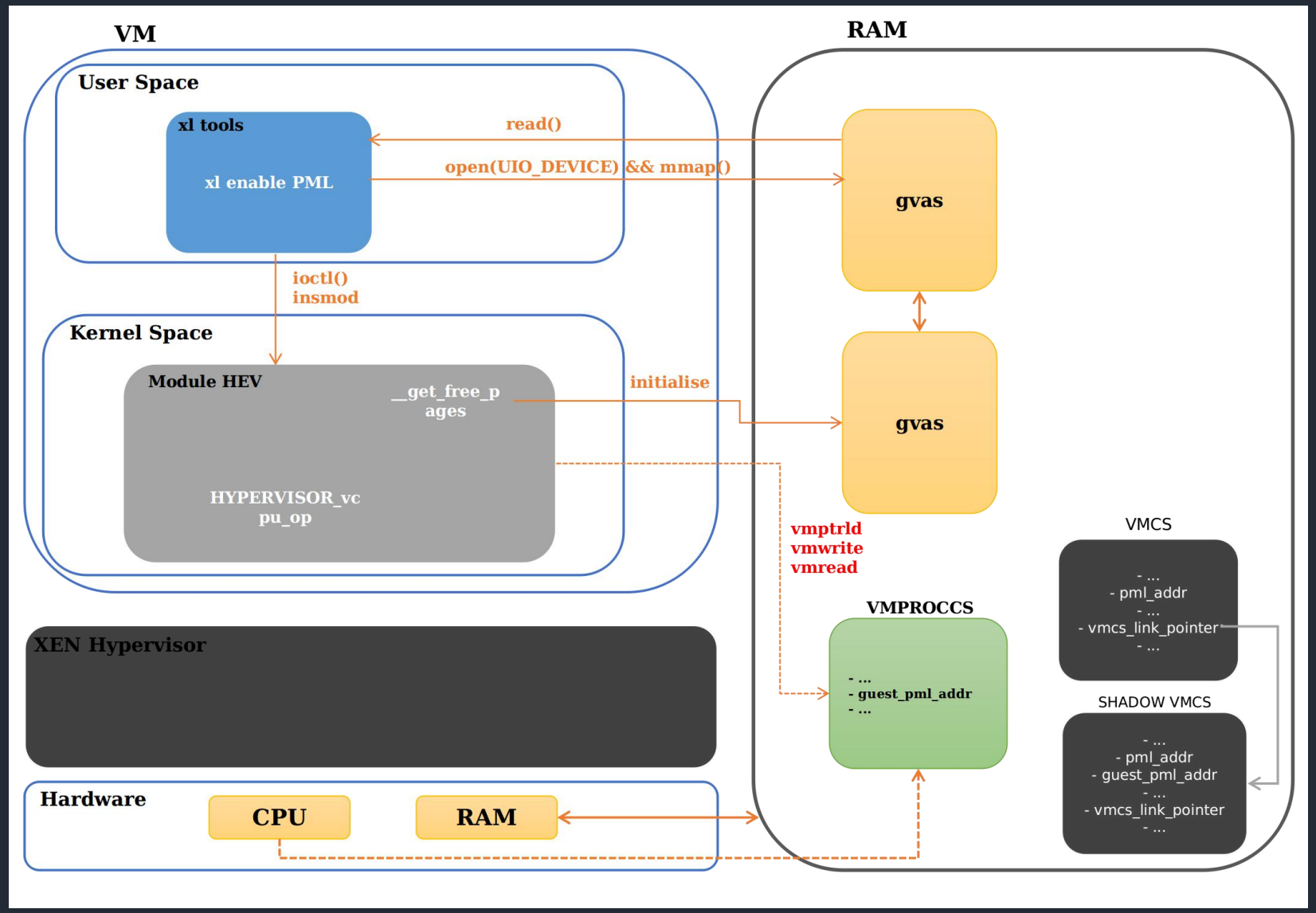
OoH: Out of Hypervisor

PML: S2 limits

- Hypervisor modification
- HW modification
- Hypercalls (only 2: for activation and deactivation of PML) ✖
- virq (virtual interrupt requests)
- Reverse mapping ✖

OoH: Out of Hypervisor

PML: S3 design



OoH: Out of Hypervisor

PML: S3 limits

- Hypervisor modification ✖
- HW modification
- Guest OS modification
- Hypercalls ✖
- virq (virtual interrupt requests) ✖
- Reverse mapping ✖

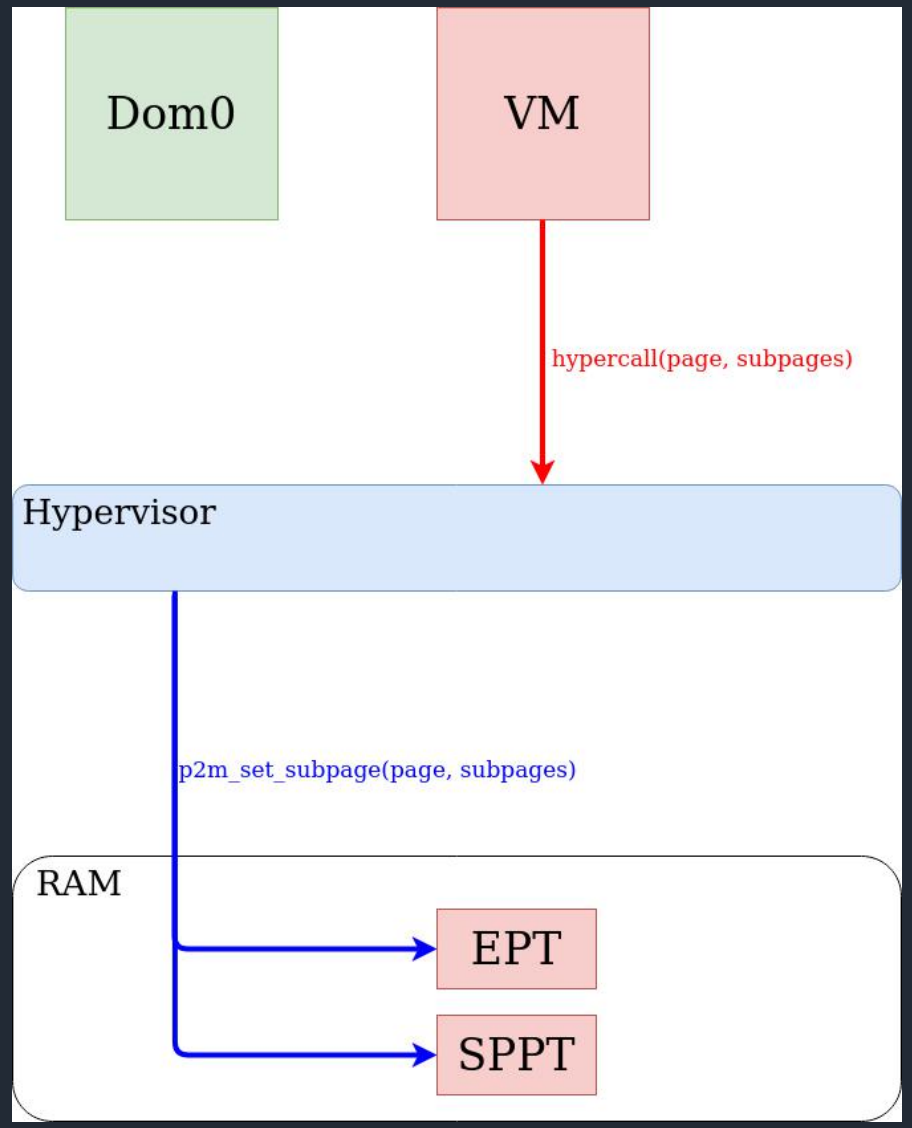
OoH: Out of Hypervisor

SPP

- 2 implementations S0 -> S1
- A kernel module and an API
- Bochs Emulator

OoH: Out of Hypervisor

SPP: S0



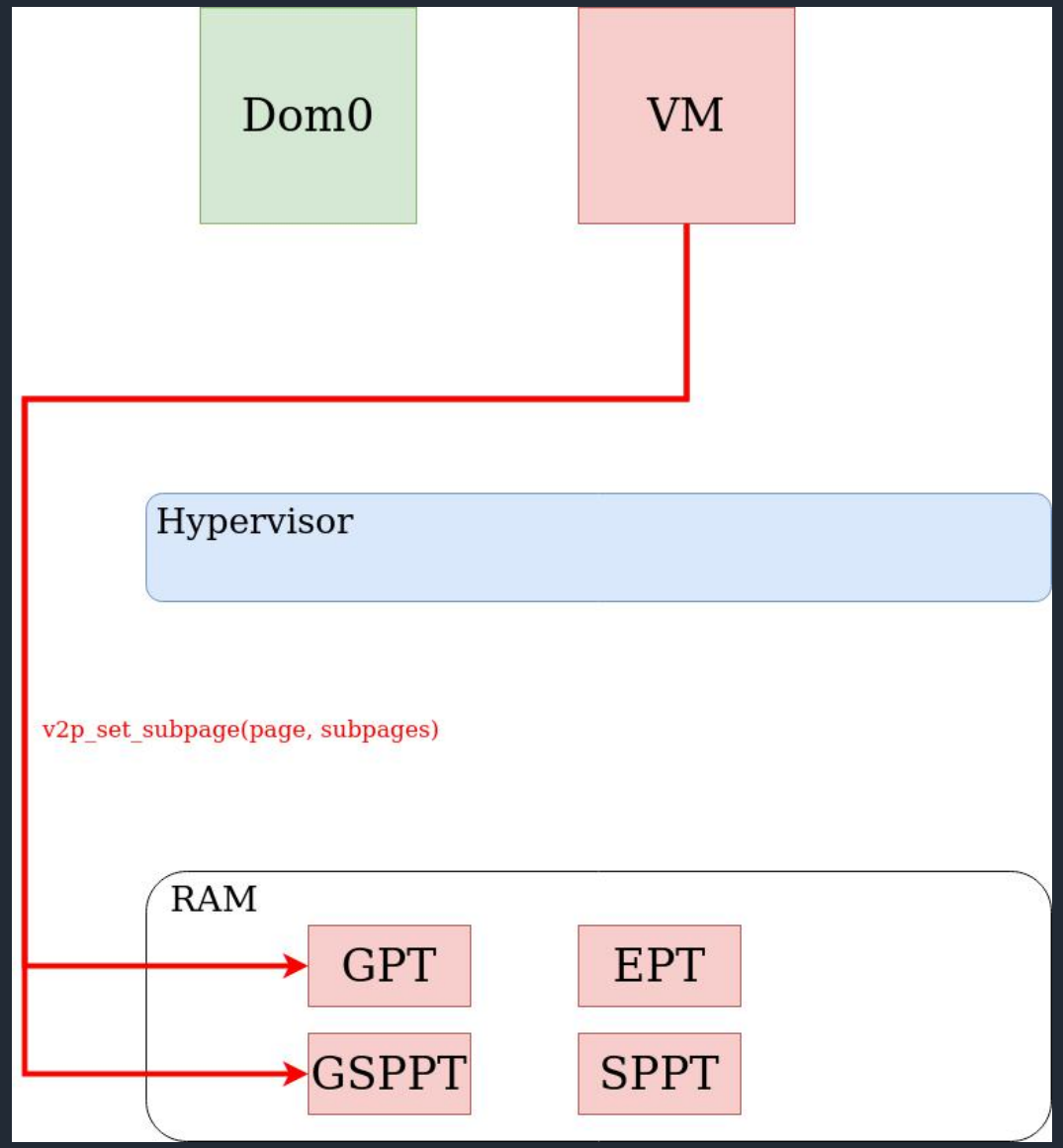
OoH: Out of Hypervisor

SPP: S0 limits

- Hypercalls
- Hypervisor modification

OoH: Out of Hypervisor

SPP: S1



OoH: Out of Hypervisor

SPP: S1 limits

- Hardware modification
- Compiler modification (new processor registers for the guest SPP Table)

OoH: Out of Hypervisor

Challenges

- Cohabitation with the hypervisor

- Cohabitation with the existing:

  - Functionnalities (shadow VMCS, SPP, PML)
  - Control structures (VMCS, SPP Table)

Conclusion

- Context: Hardware-Assisted Virtualization (HAV)

- Export HAV functionnalities to guest OS:
  - Intel PML
  - Intel SPP

Conclusion

- Context: Hardware-Assisted Virtualization (HAV)

- Export HAV functionnalities to guest OS:
  - Intel PML
  - Intel SPP

- Progress:
  - PML:
    - S0 & S1 implemented
    - S2 & S3 being implemented
    - Use case CRIU with S0

  - SPP:
    - S0 implemented
    - Use case Slimguard (a memory allocator) with S0