Updates in ACPI Based Memory Hot-Plug

Tang Chen

<tangchen@cn.fujitsu.com>

Agenda

1. ACPI & Memory Hot-Plug
2. Memory Hot-Plug Process
3. Boot Memory Handling
4. Pinned Pages Migration
5. QEmu memory Hot-Plug
6. Future work

ACPI & Memory Hot-Plug

- ACPI: Advanced Configuration and Power Interface

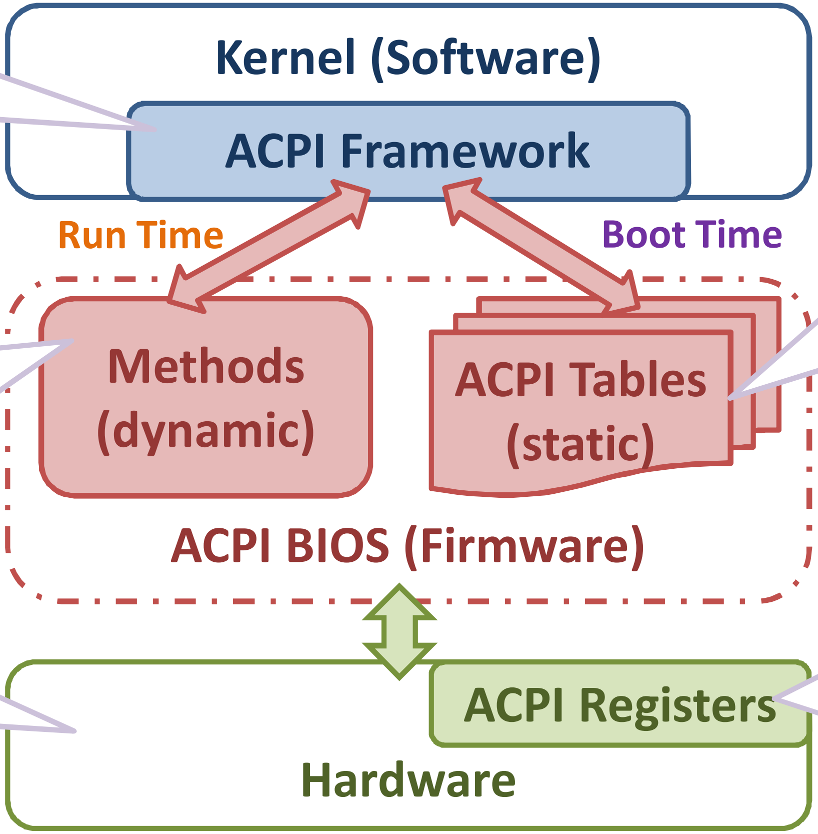
ACPI is an interface specification of Operating System-directed motherboard device configuration and Power Management.

-- ACPI Specification 5.0

OS layer framework.
- ✓ Event handling
- ✓ API

Dynamic methods used at run time.
- ✓ _EJ0
- ✓ _STA
- ✓ ……

SCI
(System Control Interrupt)



Static info used only at boot time.
- ✓ DSDT
- ✓ SRAT
- ✓ ……

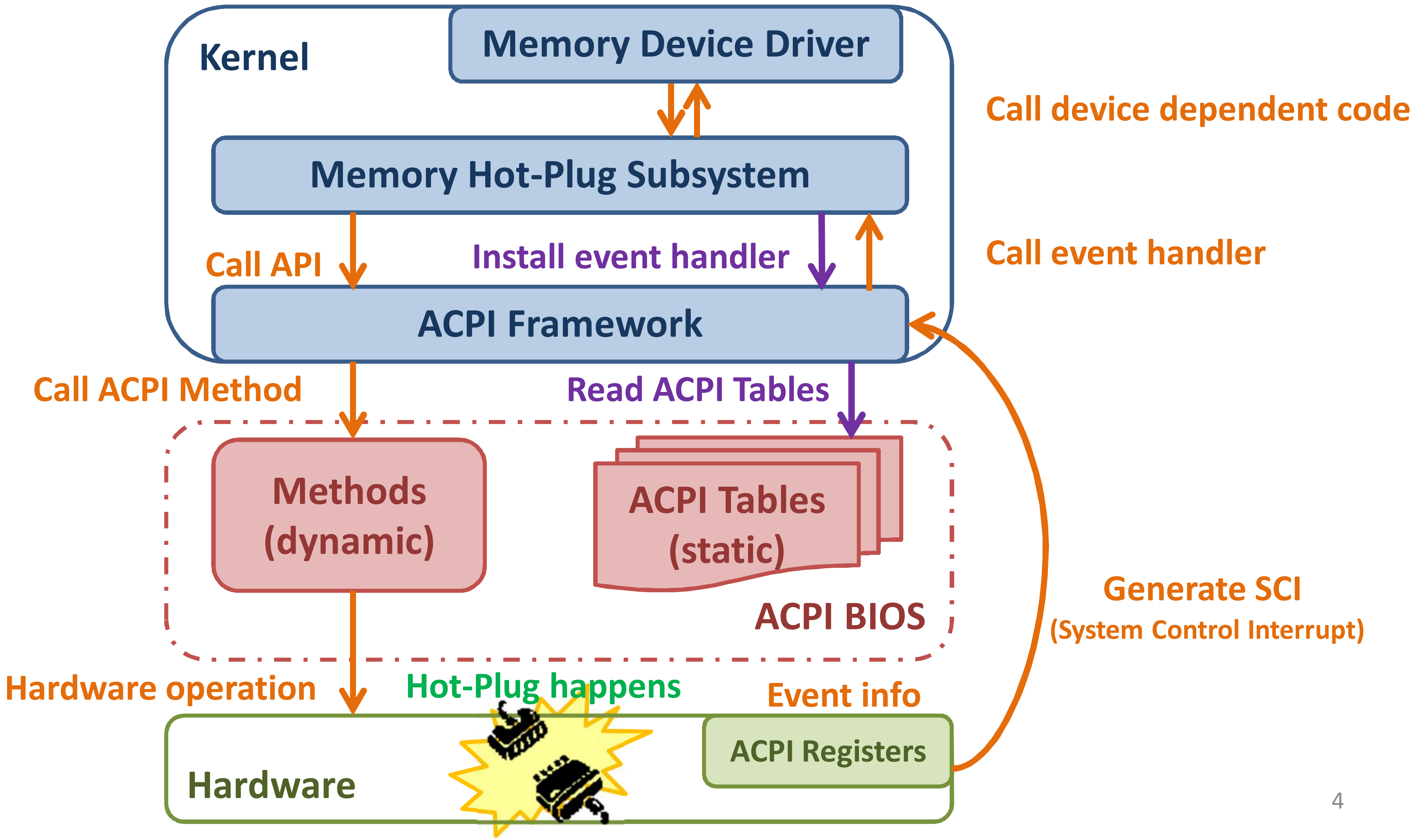
Event driven model.
- ✓ Event registers
- ✓ Control registers
- ✓ ……

ACPI & Memory Hot-Plug

- ACPI and Memory Hot-Plug

← Boot time process

← Run time process



Kernel

Memory Device Driver

Call device dependent code

Memory Hot-Plug Subsystem

Call API

Install event handler

Call event handler

ACPI Framework

Call ACPI Method

Read ACPI Tables

Methods (dynamic)

ACPI Tables (static)

ACPI BIOS

Generate SCI

(System Control Interrupt)

Hardware operation

Hot-Plug happens

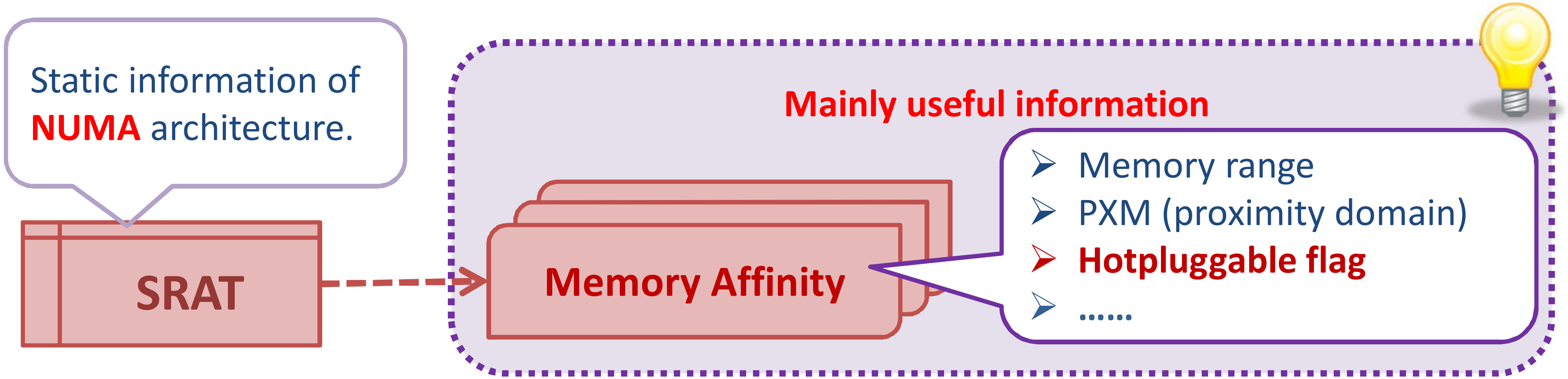
Event info

ACPI Registers

Hardware

ACPI & Memory Hot-Plug

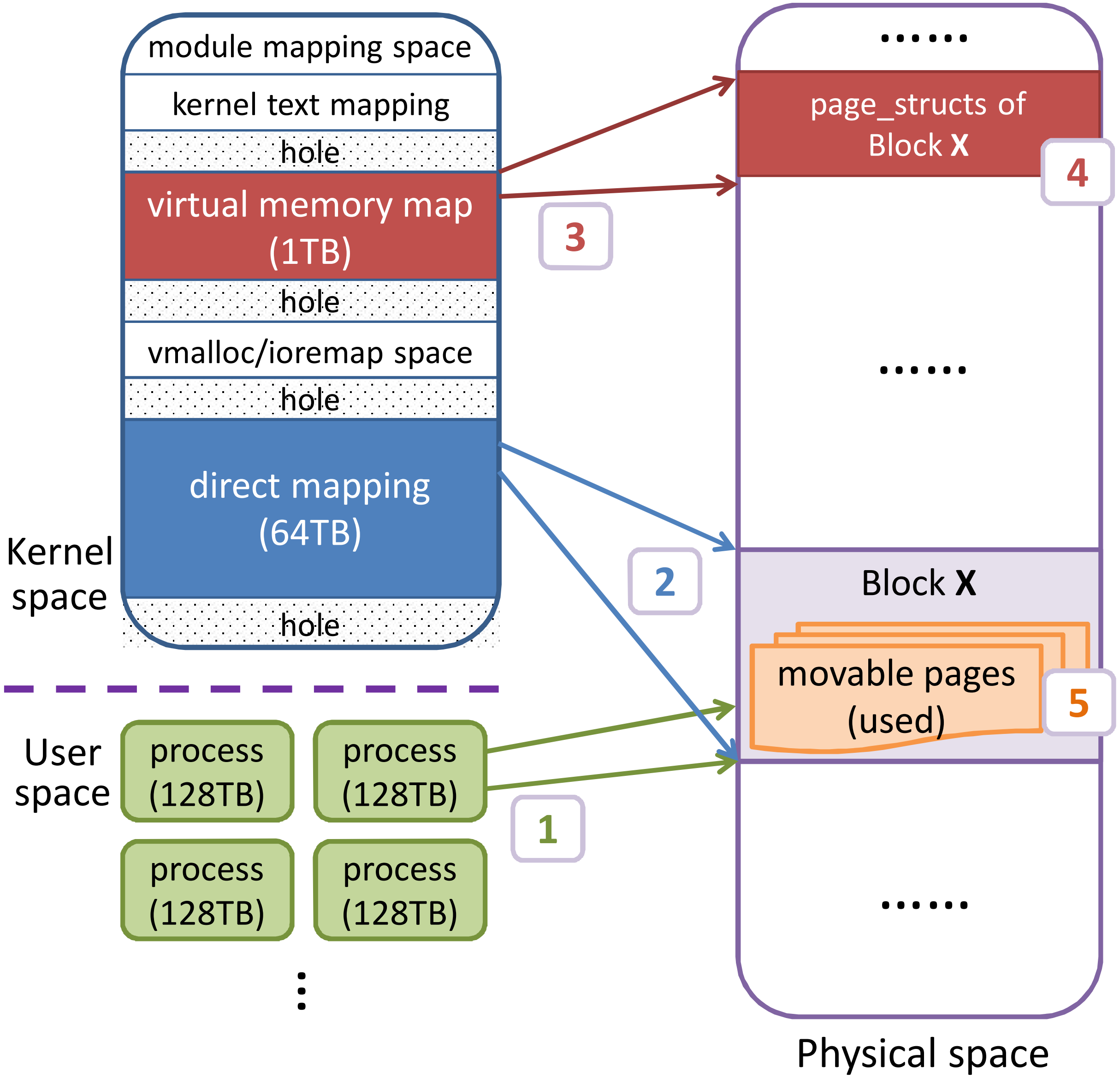
- Static configuration
  - SRAT: System Resource Affinity Table



Agenda

1. ACPI & Memory Hot-Plug
2. Memory Hot-Plug Process
3. Boot Memory Handling
4. Pinned Pages Migration
5. QEmu memory Hot-Plug
6. Future work

Memory management background



1. User processes' page tables.

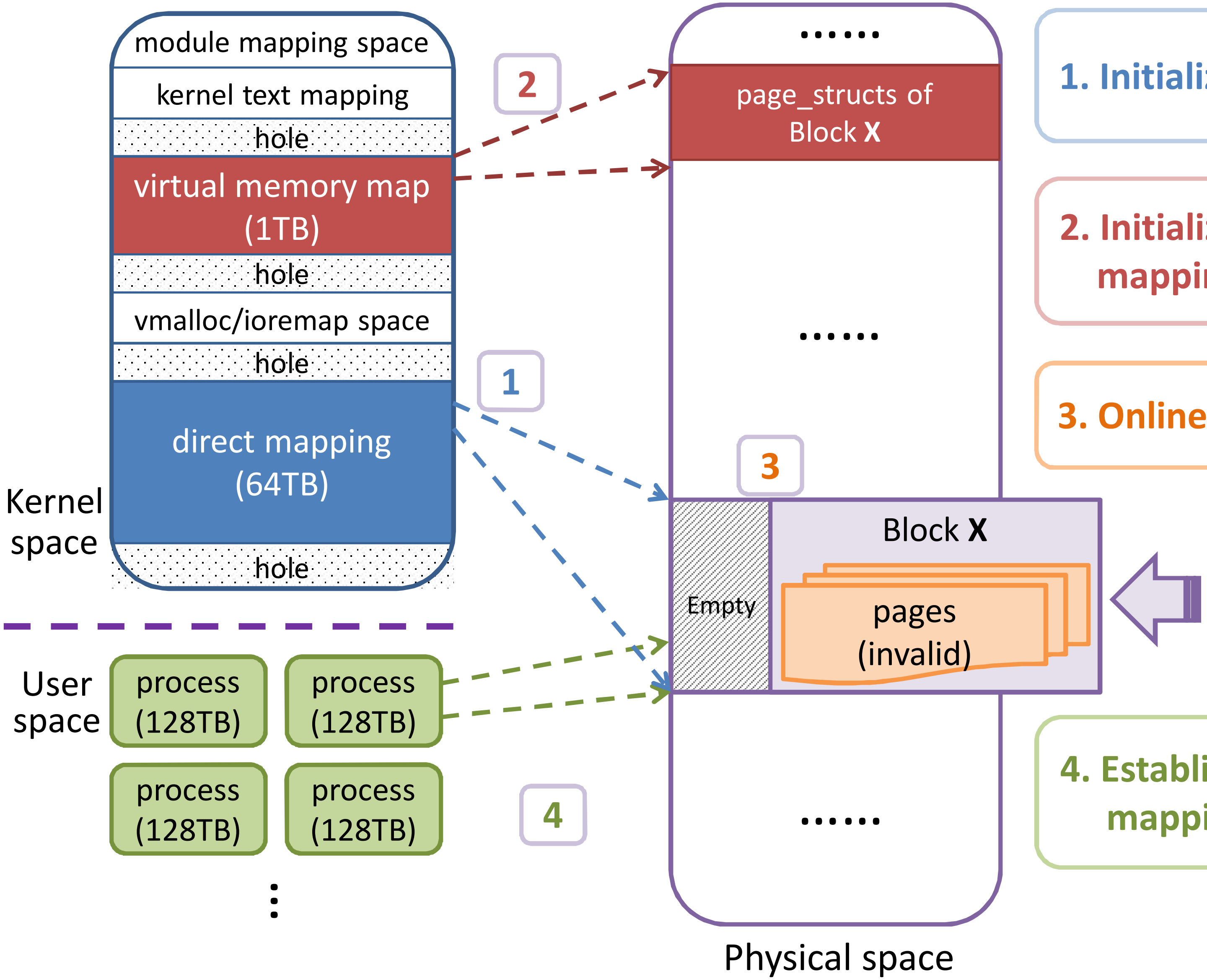
2. Kernel direct mapping page tables.

3. Virtual memory mapping page tables.

4. Virtual memory mapping pages.

5. Memory block to be hot-plugged.

Memory Hot-add Process



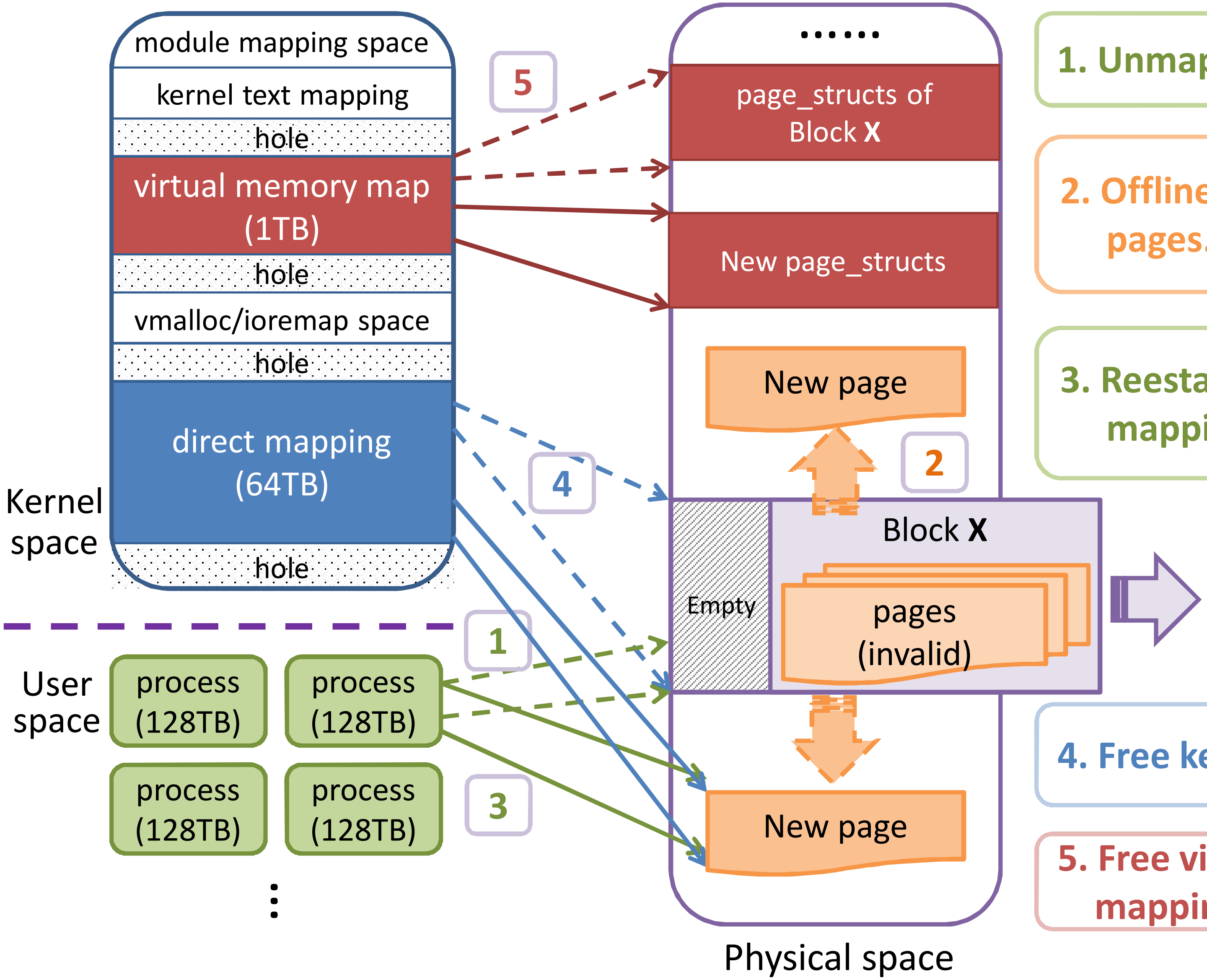
1. Initialize direct mapping.

2. Initialize virtual memory mapping.

3. Online memory block.

4. Establish user space mapping.

Memory Hot-remove Process



1. Unmap user space pages.

2. Offline pages and migrate pages.

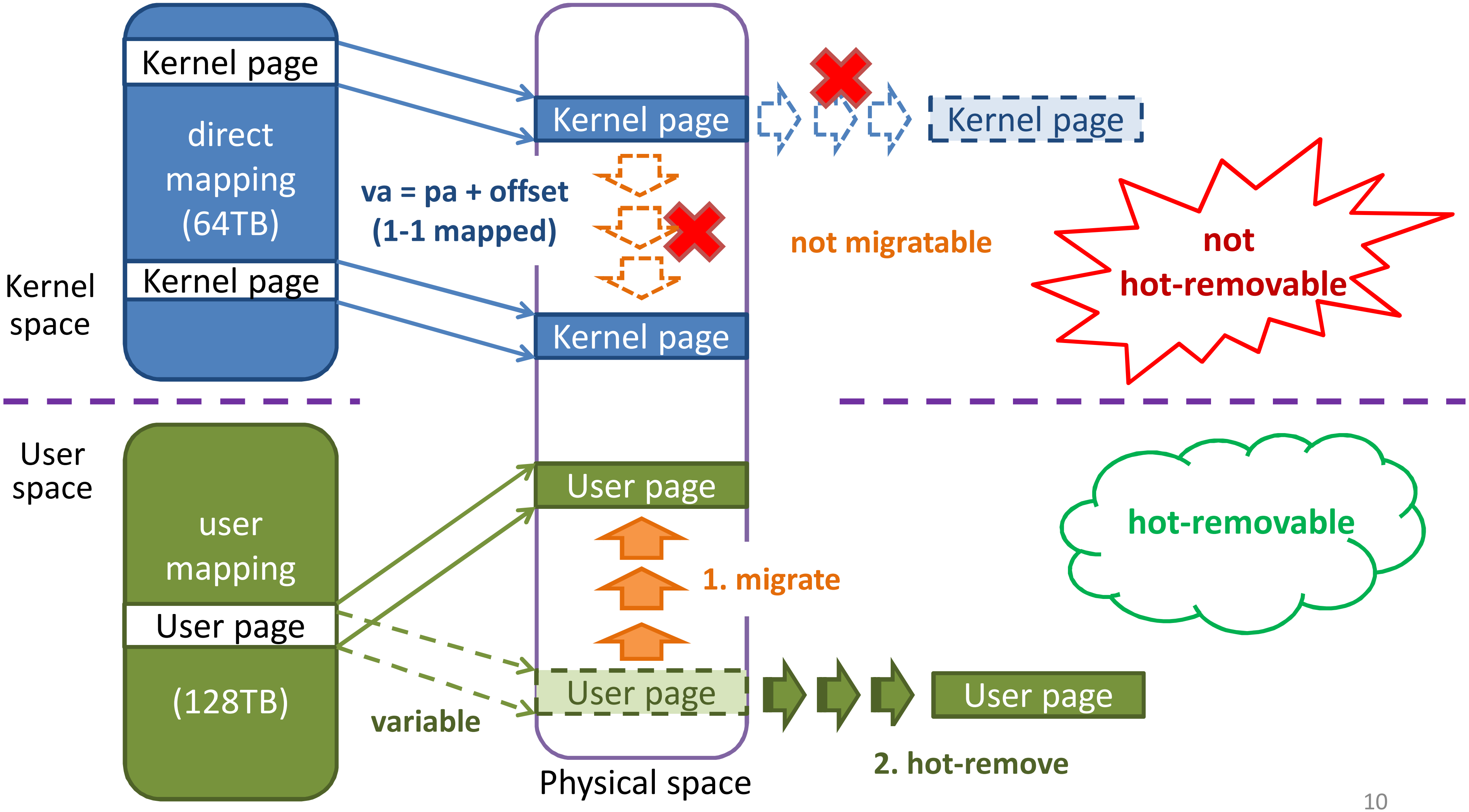
3. Reestablish new user space mapping.

4. Free kernel direct mapping.

5. Free virtual memory mapping.

Problem

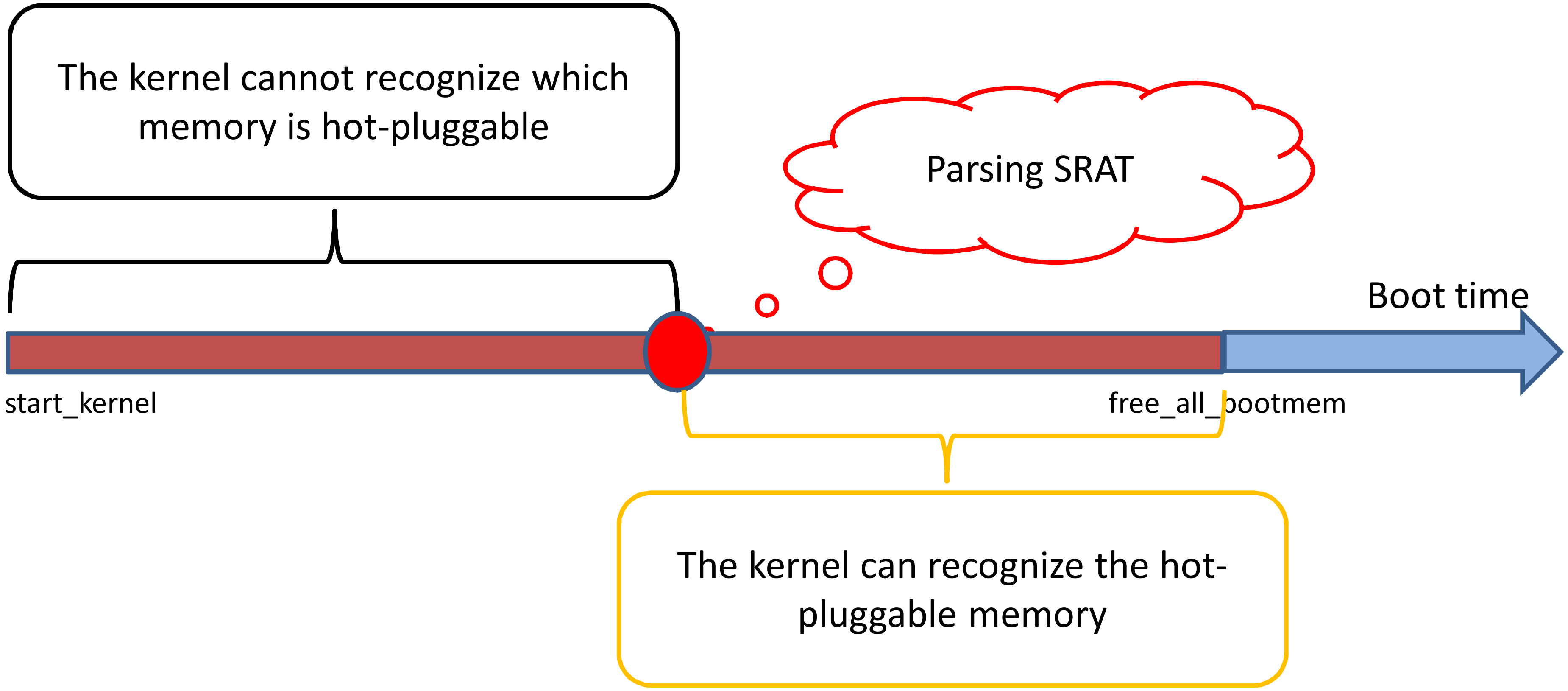
- Kernel pages cannot be hot-removed



Agenda

1. ACPI & Memory Hot-Plug
2. Memory Hot-Plug Process
3. Boot Memory Handling
4. Pinned Pages Migration
5. QEmu memory Hot-Plug
6. Future work

Avoid Allocating Hot-pluggable Memory



The kernel cannot recognize which memory is hot-pluggable

Parsing SRAT

Boot time

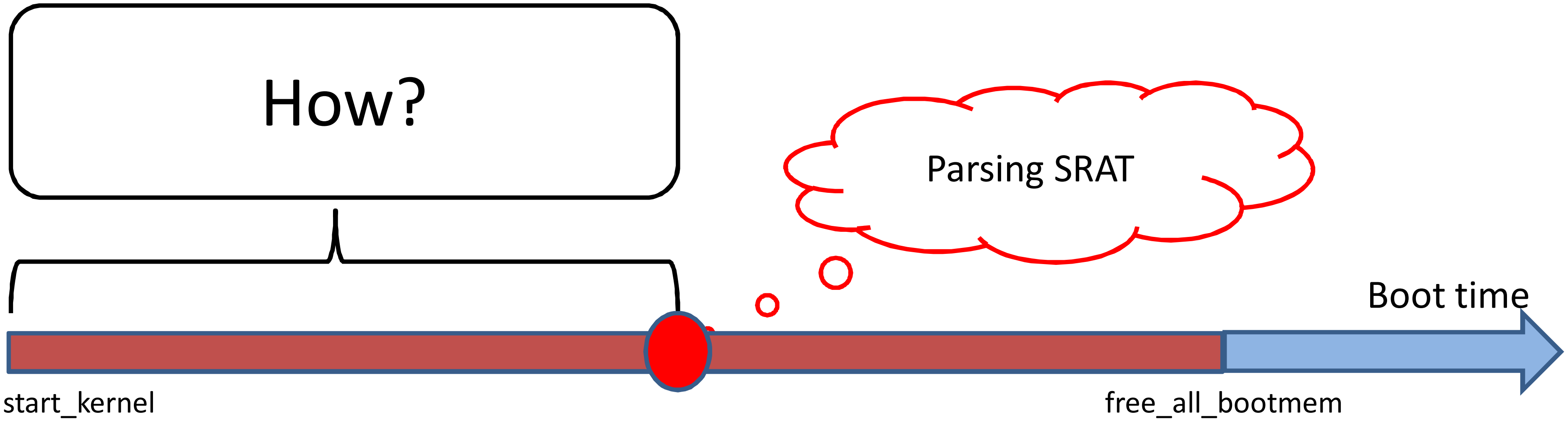
start_kernel

free_all_bootmem

The kernel can recognize the hot-pluggable memory

🟥 memblock allocator 🟦 buddy allocator

Avoid Allocating Hot-pluggable Memory
(Before Parsing SRAT)

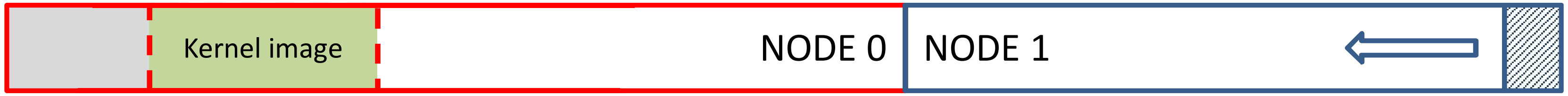


Allocate memory just behind the kernel image:

- The node kernel resides in is un-hot-pluggable
- Introduce a new bottom-up mode for memblock allocator

Top-down V.S. Bottom-up

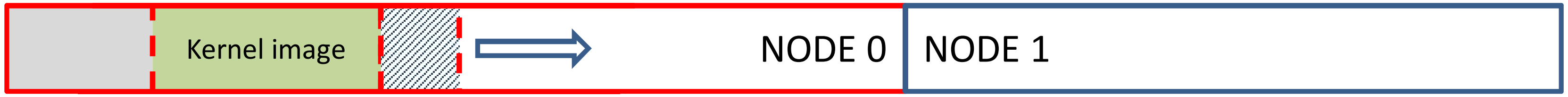
Top-down allocation mode



- Memory at low addresses is precious (e.g. for DMA devices)
- For non-memory-hot-plug users

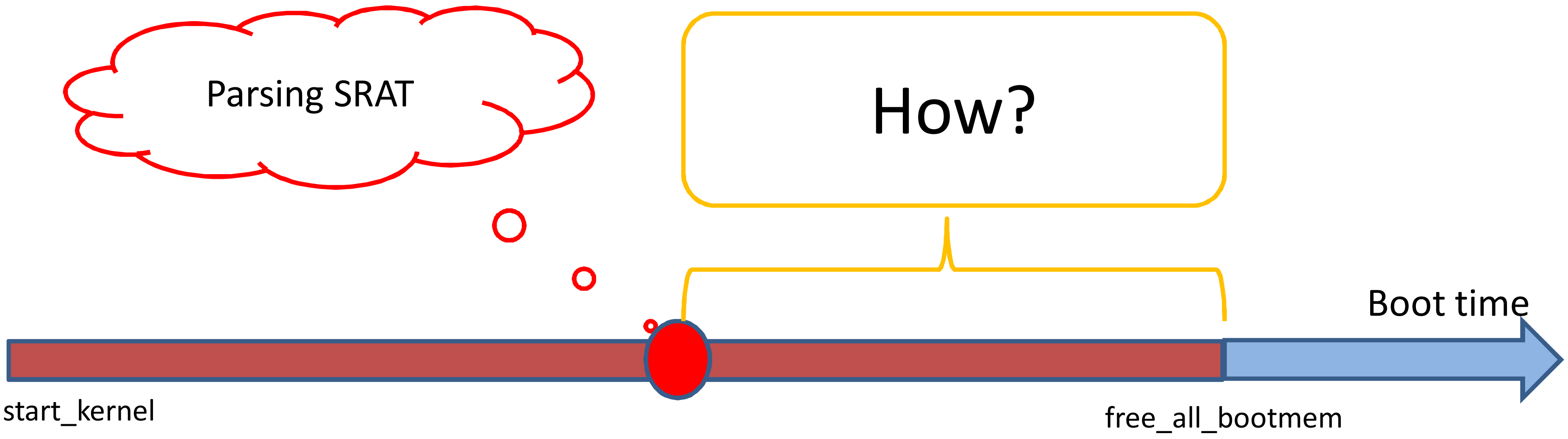
▨ new allocation

Bottom-up allocation mode



- In most cases, memory allocated before parsing SRAT won't be too much, so it could highly likely be in the same node with kernel image
- For memory-hot-plug users

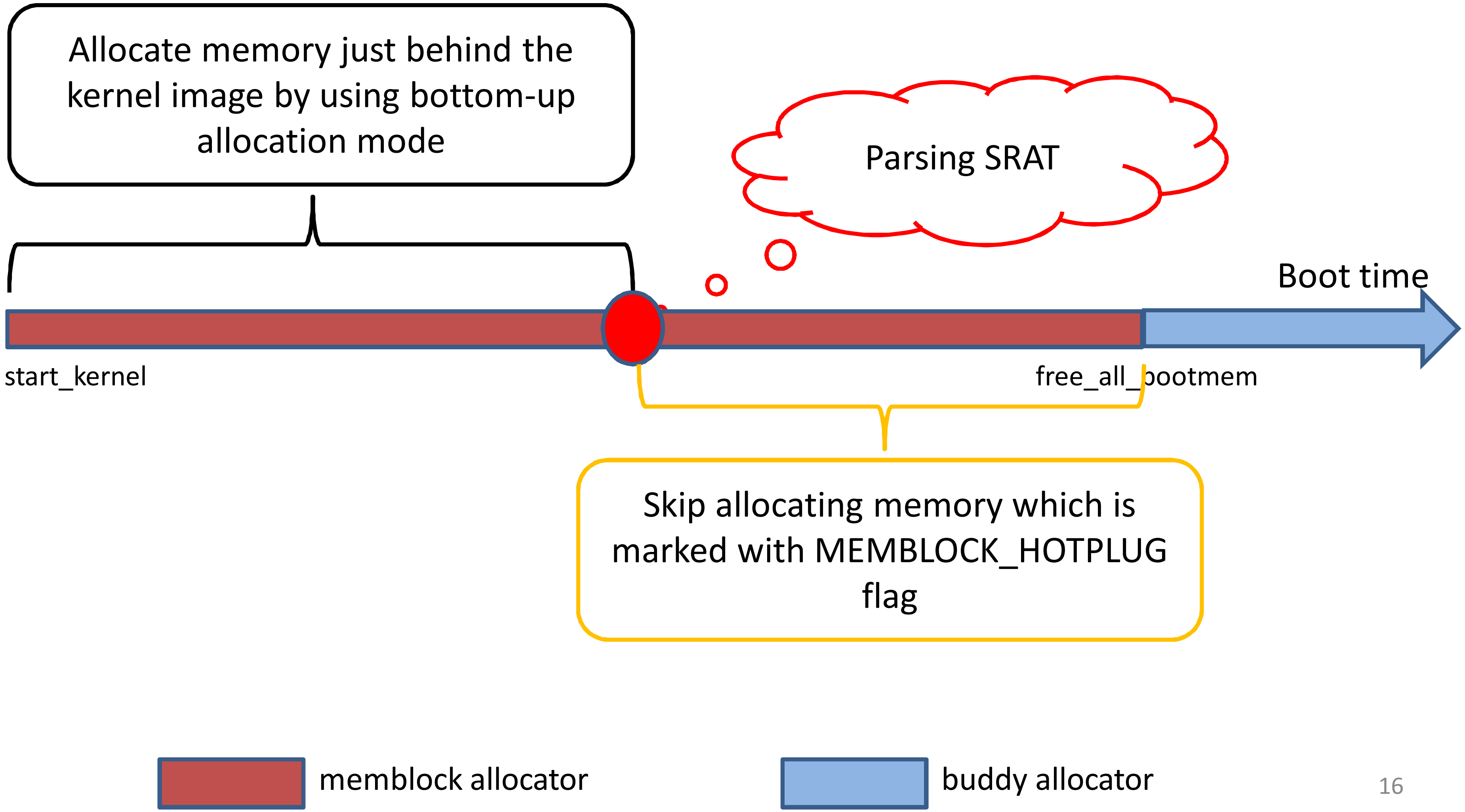
Avoid Allocating Hot-pluggable Memory
(After Parsing SRAT)



Mark hot-pluggable memory and skip them in followed allocations:

- Introduce MEMBLOCK_HOTPLUG flag for memblock allocator
- Change back to top-down mode

Summary



Allocate memory just behind the kernel image by using bottom-up allocation mode

Parsing SRAT

Boot time

start_kernel

free_all_bootmem

Skip allocating memory which is marked with MEMBLOCK_HOTPLUG flag

🟥 memblock allocator

🟦 buddy allocator

Boot Option: movable_node

- A boot-time switch to enable movable node functionality

- Higher priority than kernelcore and movablecore boot option
  - Make sure movable node functionality can be configured

Agenda

1. ACPI & Memory Hot-Plug
2. Memory Hot-Plug Process
3. Boot Memory Handling
4. Pinned Pages Migration
5. QEmu memory Hot-Plug
6. Future work

Pages Pinned by Kernel

- Short-lived Pins
  - cma, fs/exec, security, nfs, events, net/ceph, lots of dirvers…
- Long-lived Pins (pinned in all lifecycle)
  - KVM
    - Real mode identity EPT pagetable
    - APIC access page
  - AIO
    - Event Ring buffer

Short-lived Pins

- Just for data copying

- Solution: No handling
  - Memory-offline retry timeout (120s) is enough

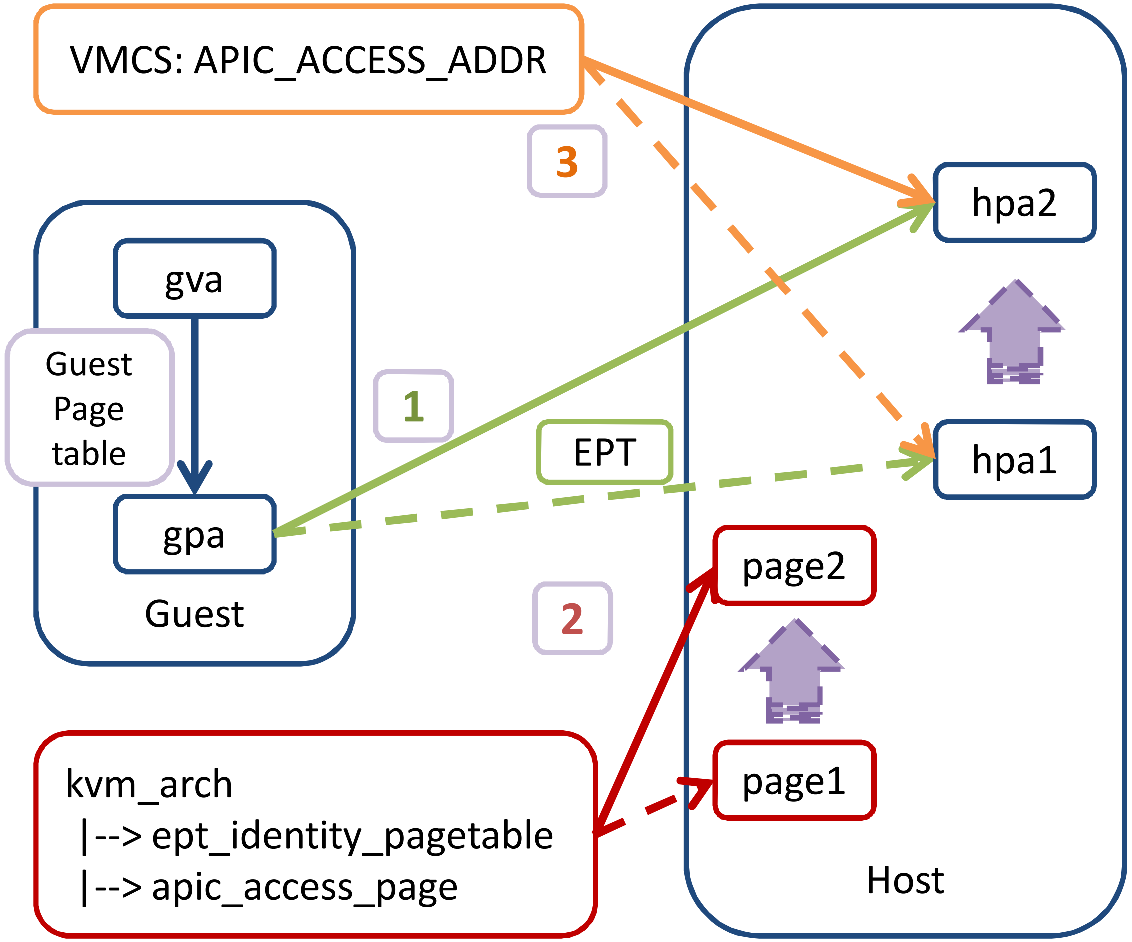
Long-lived Pins: KVM

- Real mode ept identity pagetable
  - Needed for CPUs that do not allow entering guest mode with paging disable.
  - Populated with ptes that cover entire guest's memory.

- APIC access page
  - Used by CPU directly to catch MMIO access to an APIC.

Long-lived Pins: KVM

Why pinned: For convenience, not necessary.



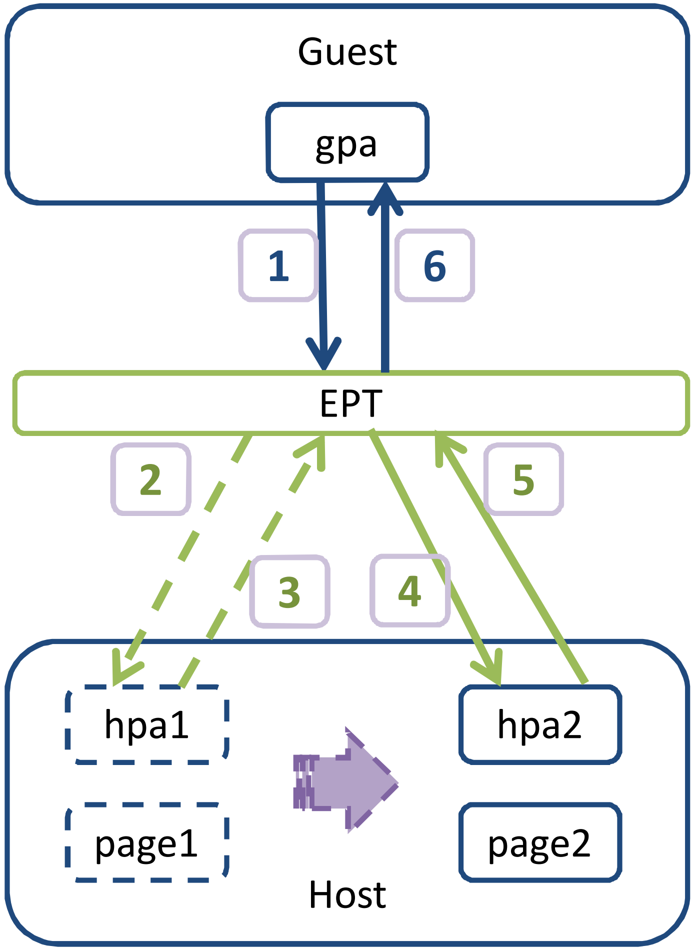
1. Update EPT entry.

2. Update local cache for page struct.

3. Update APIC_ACCESS_ADDR pointer in VMCS.

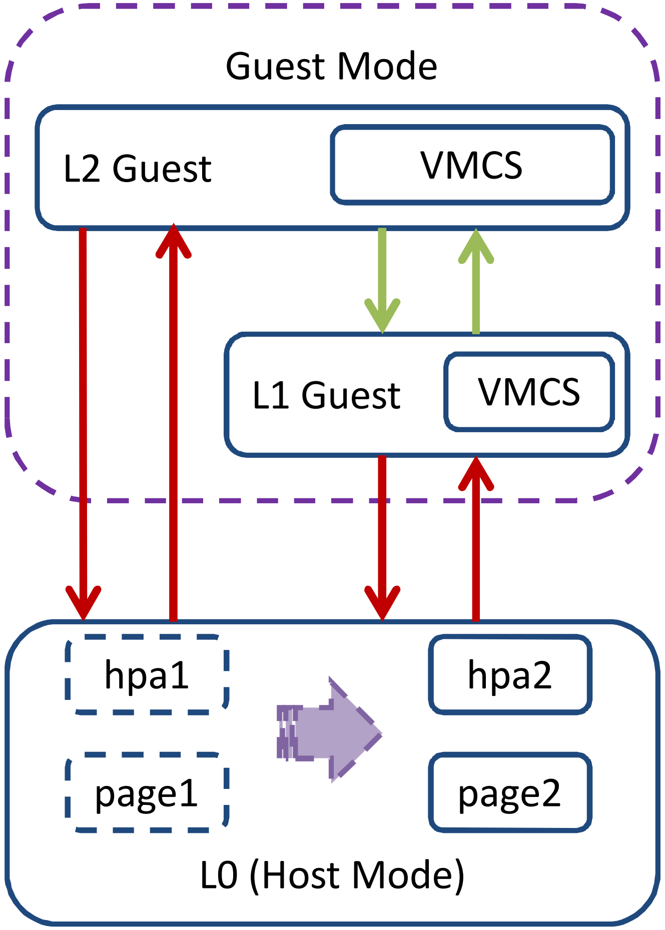
Long-lived Pins: KVM

- EPT identity pagetable: Unpin directly



1. Guest requires a page
2. MMU searches EPT for a hpa
3. MMU returns EPT violation since page has been migrated
4. KVM handles EPT violation and find the new page
5. KVM updates EPT
6. MMU returns new hpa

Long-lived Pins: KVM APIC access page



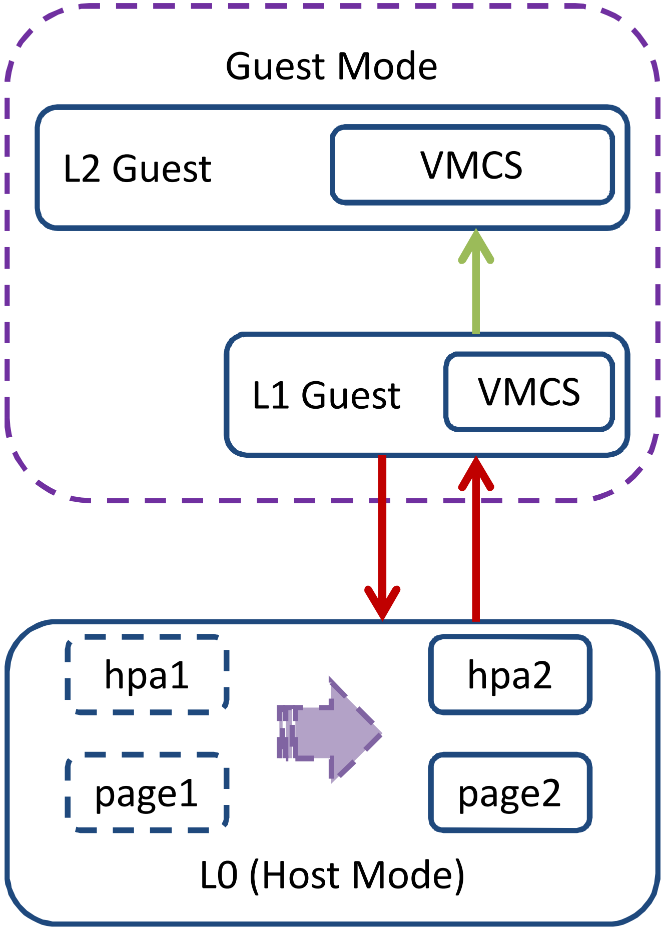
Two kinds of VM-Entry/Exit:

- **VM-Entry/Exit in Guest mode are emulated by KVM.**
  - **L1 <--> L2 VM-Entry/Exit**
- **VM-Entry/Exit between Host and Guest mode are provided by CPU.**
  - **L0 <--> L1 VM-Entry/Exit**
  - **L0 <--> L2 VM-Entry/Exit**

Two cases to handle:

- **CPU is running L1 Guest.**
- **CPU is running L2 Guest.**

Long-lived Pins: KVM APIC access page



CPU is running L1 Guest:

Page Migration:

1. Try to unmap page
2. MMU notifier works
3. Unmap and migrate page

……

4. Page migrated

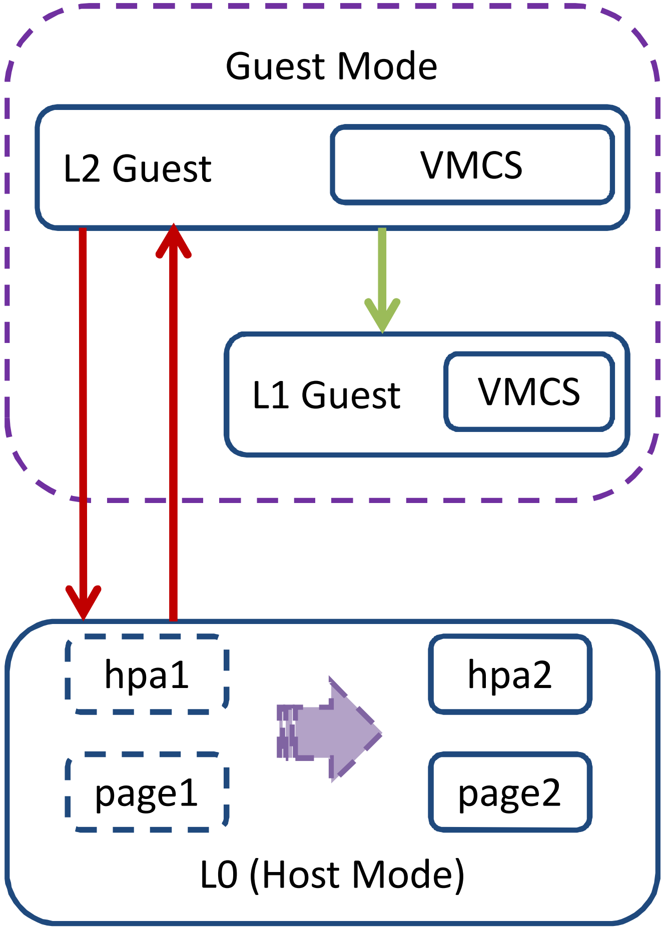
KVM:

1. Handler enforces a L1 --> L0 VM-Exit
2. Handler makes vcpu request to update L1 VMCS
3. Next L0 --> L1 VM-Entry: GUP waits for page migration

……

4. Update L1 VMCS
5. Update L2 VMCS in next L1 --> L2 VM-Entry

Long-lived Pins: KVM APIC access page



CPU is running L2 Guest:

Page Migration:

1. **Try to unmap page**
2. **MMU notifier works**
3. **Unmap and migrate page**

……

4. **Page migrated**

KVM:

1. **Handler enforces a L2 --> L0 VM-Exit**
2. **Handler makes vcpu request to update L2 VMCS**
3. **Next L0 --> L2 VM-Entry: GUP waits for page migration**

……

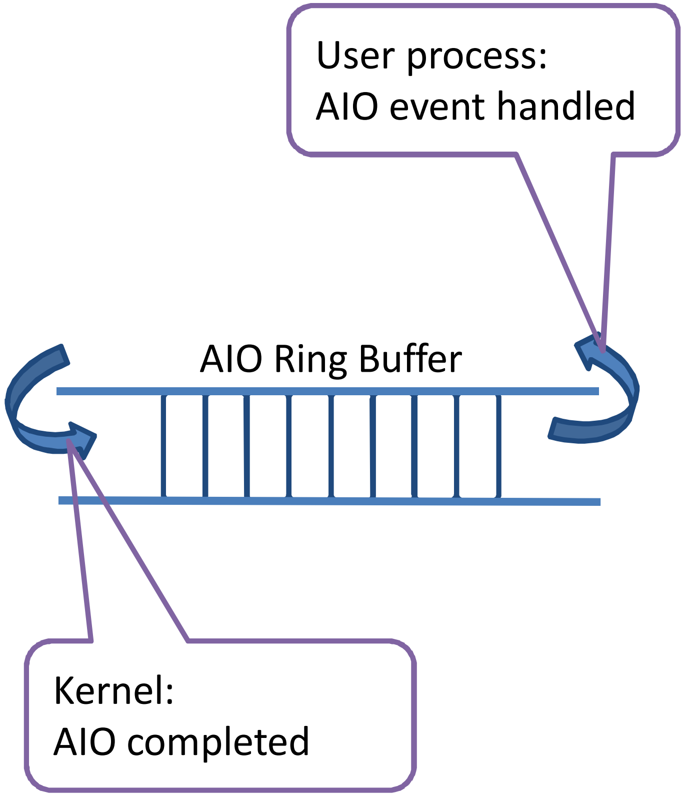
4. **Update L2 VMCS**
5. **Update L1 VMCS in next L2 --> L1 VM-Exit**

Long-lived Pins: AIO

- AIO Event Ring Buffer
  - Used by kernel to notify user space that AIO has completed.

Long-lived Pins: AIO

Why pinned: Unable to know when AIO completes.



Page Migration:

1. Offline memory
2. Page migration
   |--> migratepages()

   ……

3. Page migration fails

AIO:

1. AIO pins ring pages
2. AIO in progress

   ……

3. AIO completes
4. AIO unpin ring pages

Long-lived Pins: AIO

Need new splution

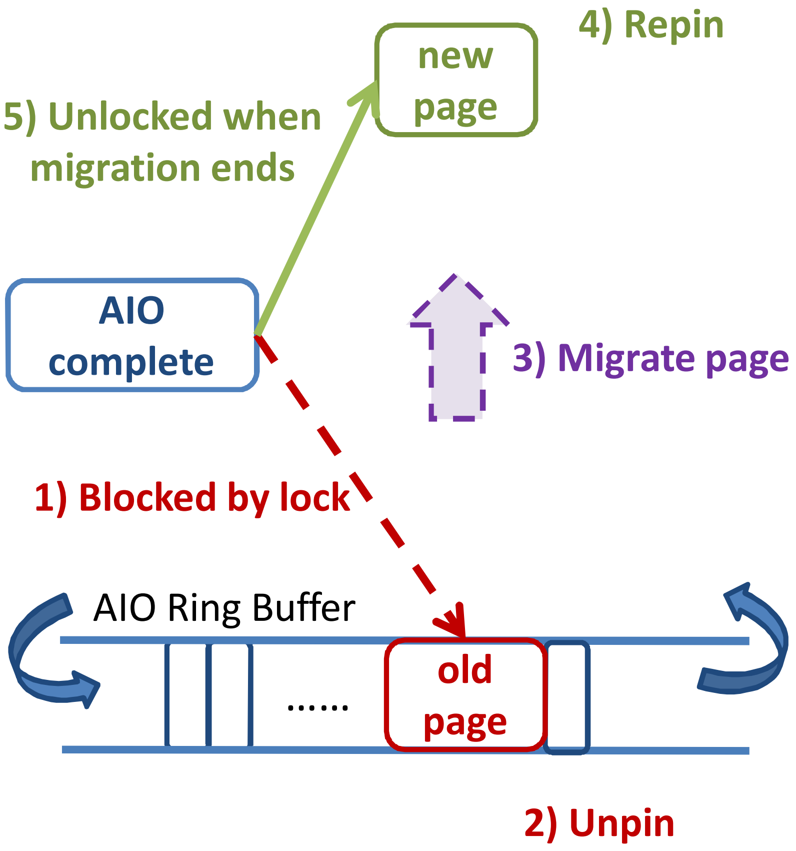
Why cannot use MMU notifier:

1. **No way to get the page (have to repin)**
   - **GUP may sleep in io interrupt context**

2. **No way to notify AIO to repin the page**
   - **No such MMU notifier after page migration completes**

Long-lived Pins: AIO

Solution: Implement aio_migratepage()



Page Migration:

1. Offline memory
2. Page migration starts

……

3. Page migration in progress

……

4. Page migration ends

AIO:

1. AIO pins ring pages
2. AIO in progress

……

3. aio_migrate()
   1) lock
   2) unping ring pages
   3) migrate ring pages

……

   4) repin ring pages
   5) unlock

……

4. AIO completes
5. AIO unpin ring pages

Agenda

1. ACPI & Memory Hot-Plug
2. Memory Hot-Plug Process
3. Boot Memory Handling
4. Pinned Pages Migration
5. QEmu memory Hot-Plug
6. Future work

QEmu memory hotplug

Memory hot-add usage: available

- QEmu commandline:

```
-m 2G,slots=8,maxmem=16G
-object memory-ram,id=ram0,size=1G
-object memory-backend-file,mem-path=/hugetlbfs,id=ram1,size=1G
```

- QEmu monitor:

```
device_add pc-dimm,id=d0,memdev=ram0
object_add memory-ram,id=ram2,size=2G
object_add memory-backend-file,mem-path=/hugetlbfs,id=ram3,size=1G
```

Memory hot-remove usage: in progress

- QEmu monitor:

```
device_del d0
```

Agenda

1. ACPI & Memory Hot-Plug
2. Memory Hot-Plug Process
3. Boot Memory Handling
4. Pinned Pages Migration
5. QEmu memory Hot-Plug
6. Future work

Future work

- Try to migrate kernel pages
  - Long way to go.
- QEmu device hotplug
  - CPU hotplug
  - Device hotplug framework improvment .
- User space tools, like libnuma and numactl
  - A library of functions.
  - Commands.

Thank you!
Q&A