Universitatea
POLITEHNICA
din București

Migrating a bhyve guest

BSDCan2019, Ottawa, Canada

Authors

Elena Mihăilescu
elenamihailescu22@gmail.com

Mihai Carabaș
mihai.carabas@cs.pub.ro
mihai@freebsd.org

About me

- Master's degree student at University POLITEHNICA of Bucharest
- Study Complex Network Security

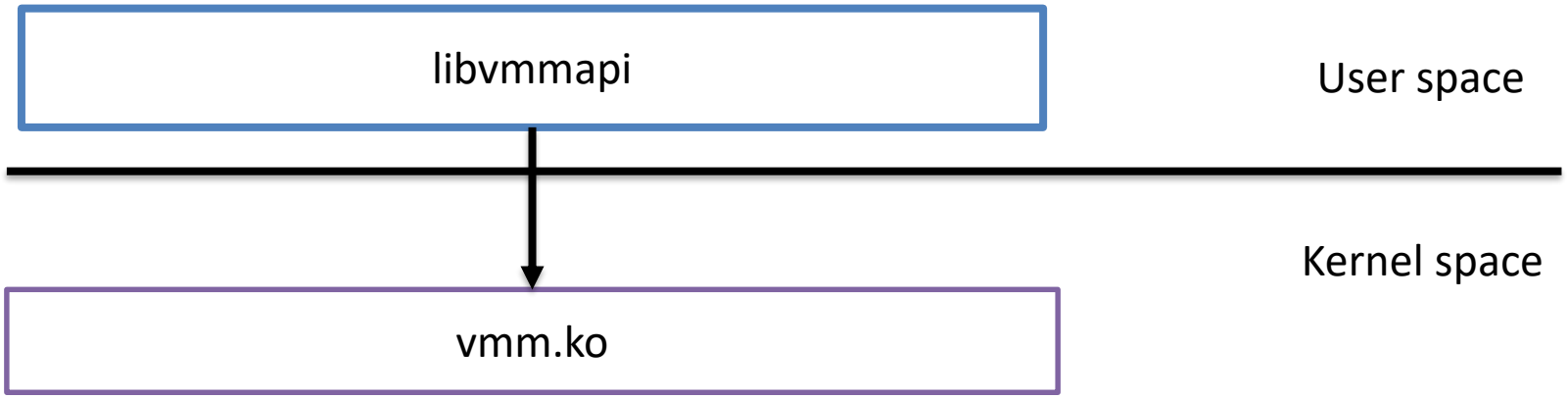
- Working on FreeBSD's projects since September, 2017

Introduction

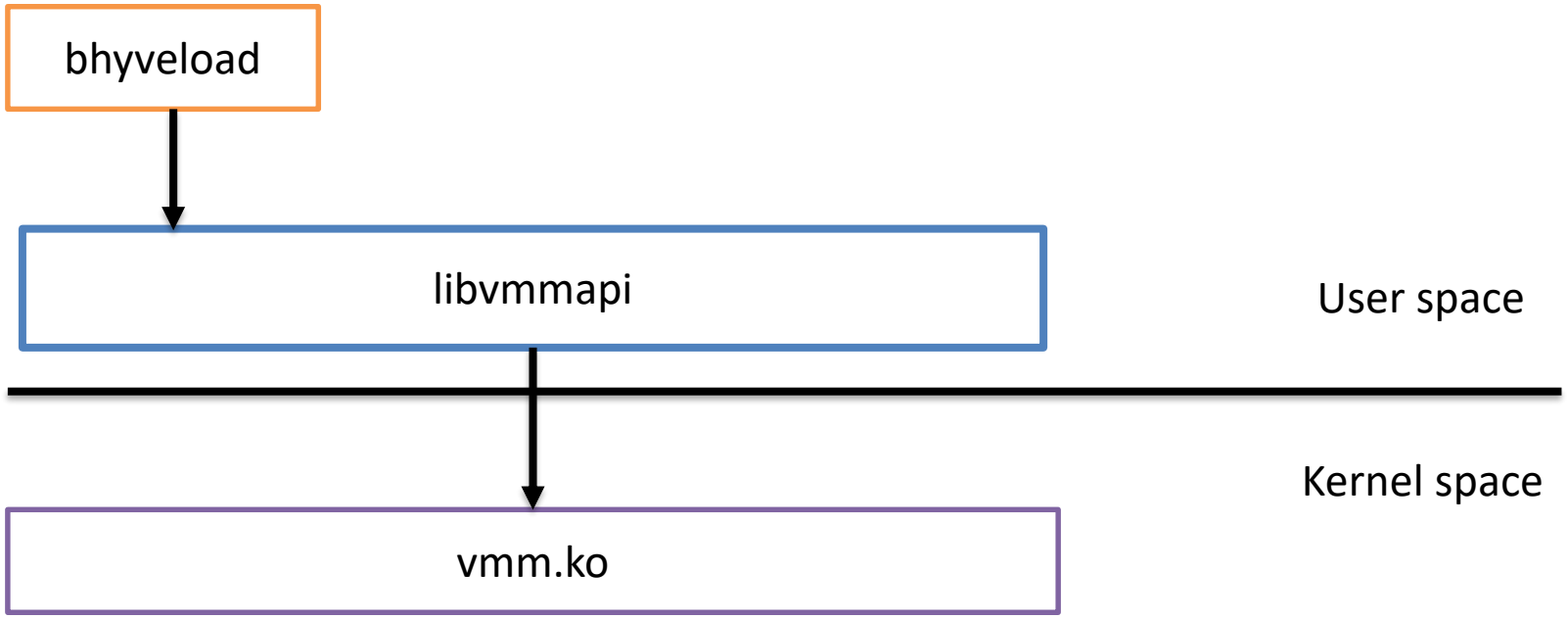
- Virtualization & Cloud Solutions
- Live Migration

- XEN, Hyper-V, KVM, VirtualBox, VMWare
- bhyve – FreeBSD's hypervisor

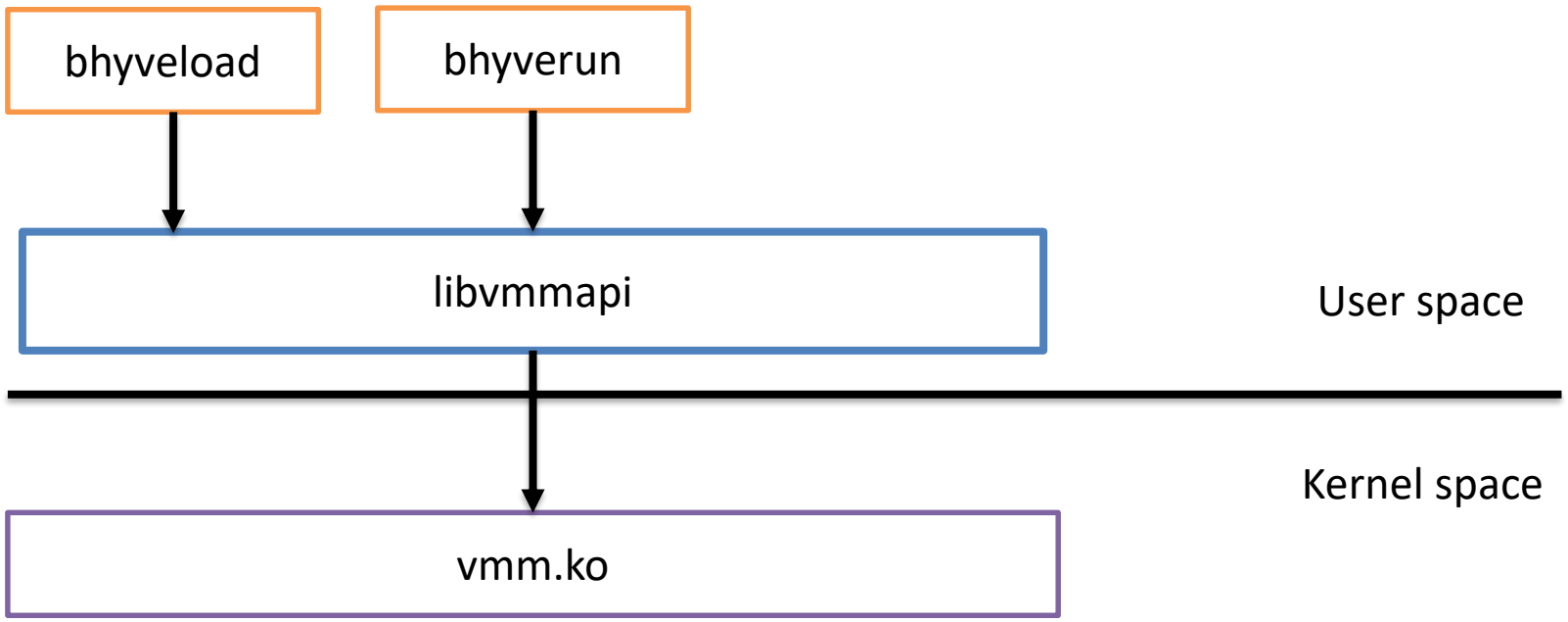
bhyve



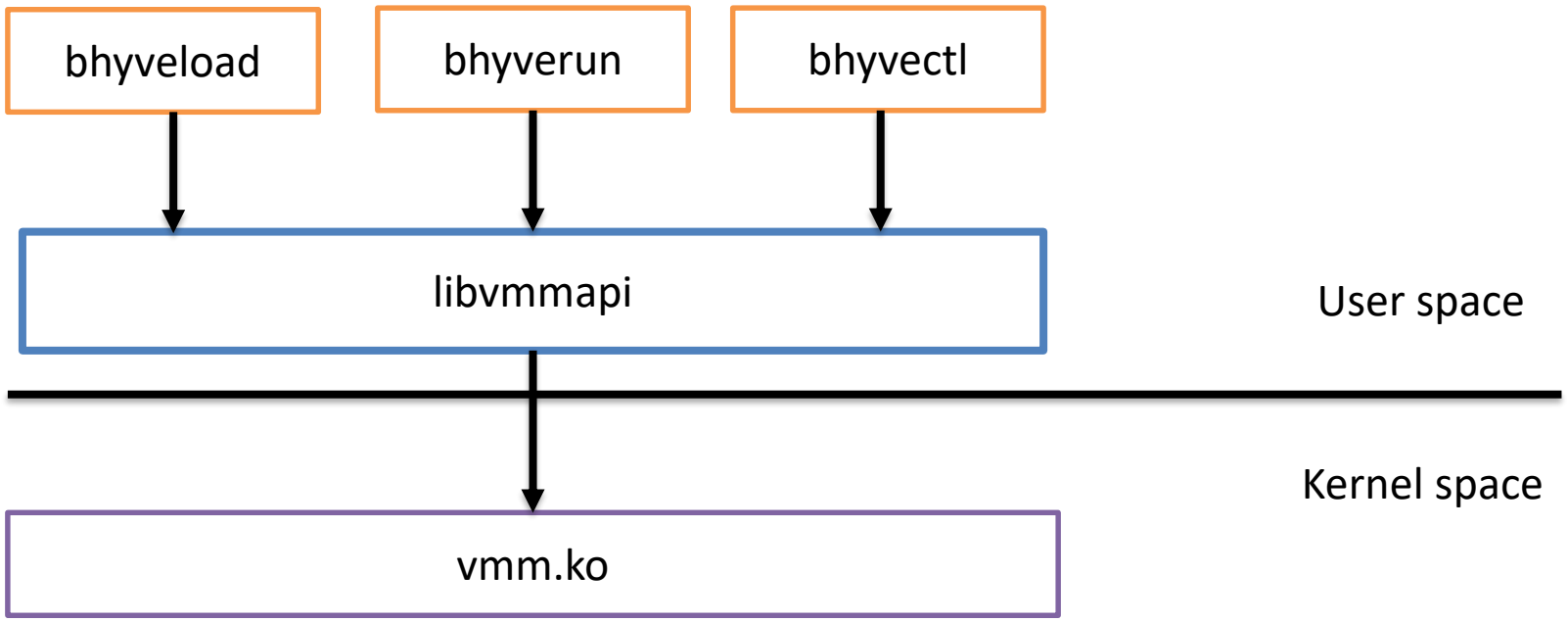
bhyve



bhyve


bhyveload
bhyverun
libvmmapi
User space
Kernel space
vmm.ko

bhyve


bhyveload
bhyverun
bhyvectl
libvmmapi
User space
Kernel space
vmm.ko

Virtual Machine Migration

- Move a guest from one host to another

- Cold Migration
- Warm Migration
- Live Migration
  - Pre-Copy Live Migration
  - Post-Copy Live Migration

Types of Migration

Cold Migration

- Guest is powered off
- Move its disk on another system

- Disadvantages:
  - Process is really slow (big down time)
  - The guest has to be powered off

Types of Migration

Warm Migration

- Guest is suspended
- Transfer its state and memory on another host
- Resume guest on destination system
- Guest disk image has to be shared

- Disadvantages:
  - Big downtime (i.e., large sized guests)

Types of Migration

Live Migration

- Guest memory is migrated while running
- At some point, guest is suspended and only the CPU's & devices' state is migrated
- Short down time

Types of Migration

Live Migration – Pre-Copy Approach

- Memory migrated in rounds while guest is running
- In final round:
  - Stop source VM
  - Copy remaining memory
  - Copy CPU & devices state
  - Start destination VM

Types of Migration

Live Migration – Post-Copy Approach

- Memory migrated using a page fault approach
- Algorithm:
  - Stop source VM
  - Copy CPU and devices state on destination
  - Start destination VM
  - Copy memory page when a page fault occurs at destination

Types of Migration

Pre-Copy Live Migration

- Same page can be migrated multiple times
- Guest running on source until migration finishes
- If migration fails, guest continue running on source host

Post-Copy Live Migration

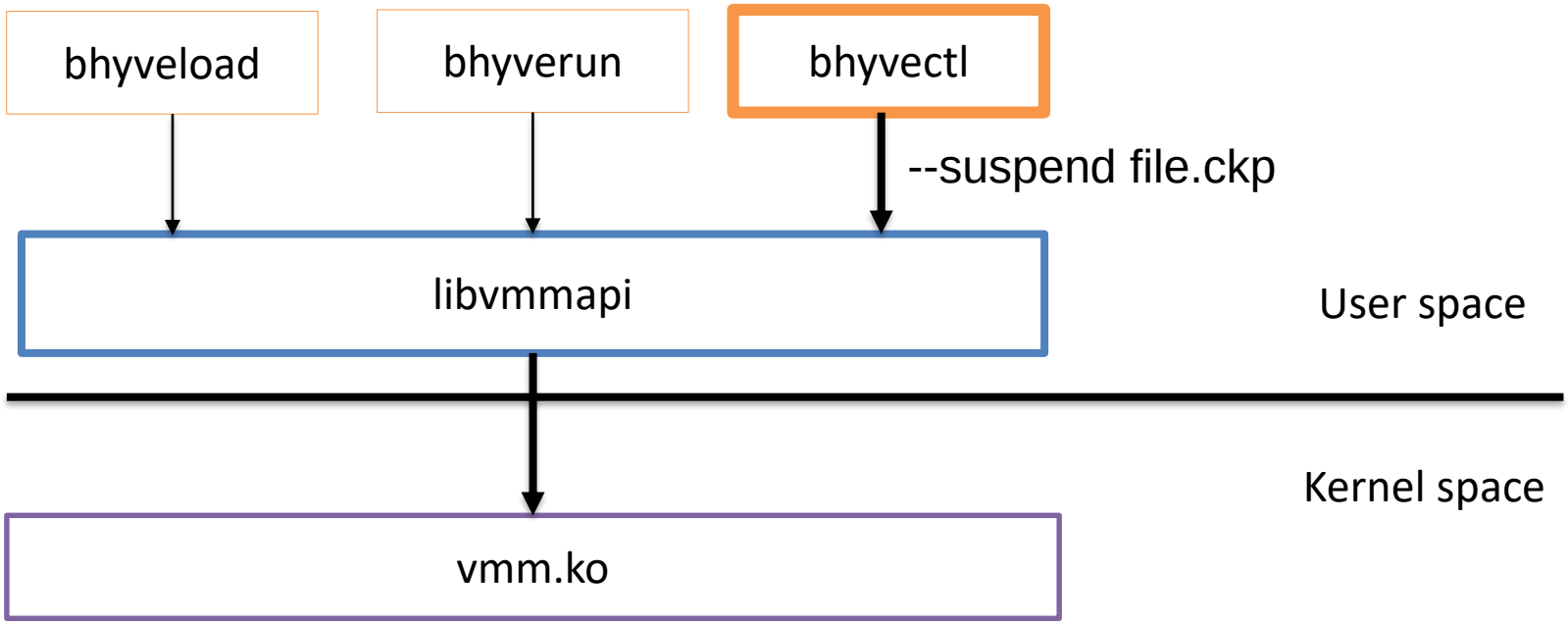
- A page is migrated only once
- Guest running on destination until migration finishes
- If migration fails, additional mechanism should be implemented for fallback

Related Work

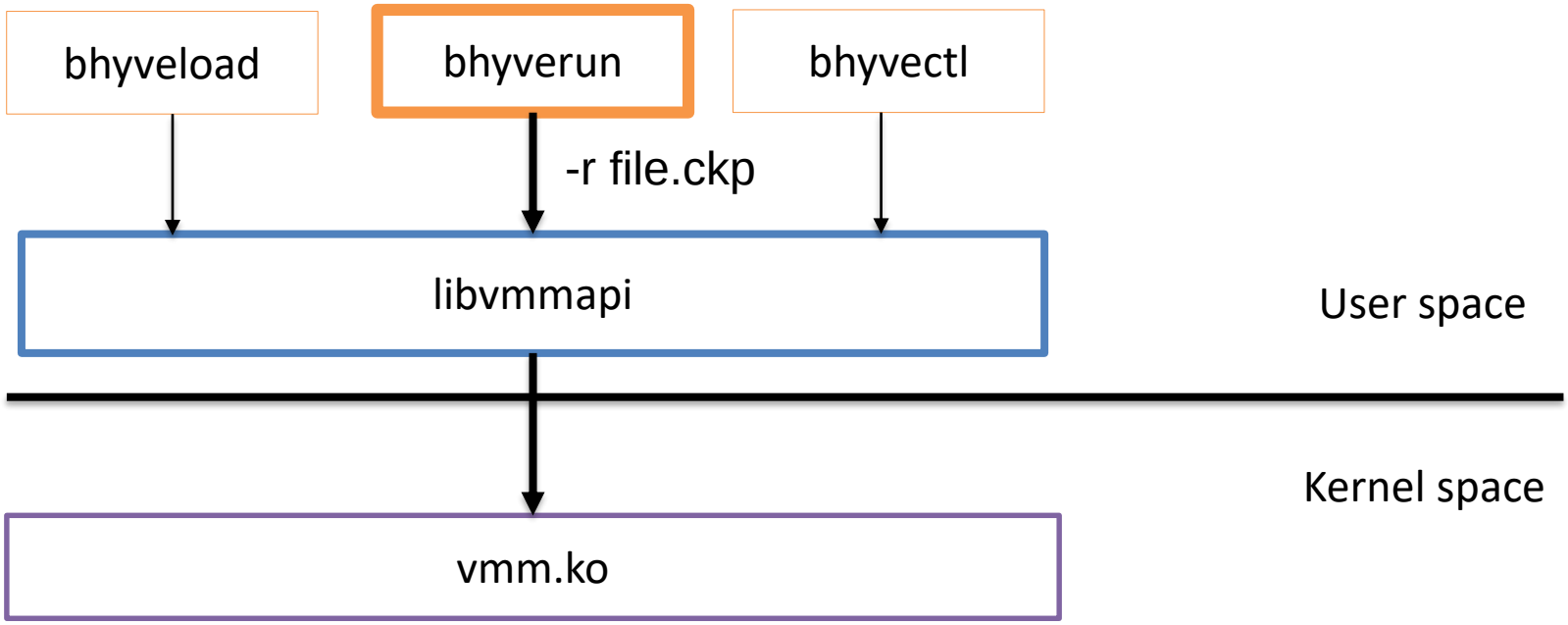
Save and Restore feature for bhyve

- Intel/AMD CPU state – VMCS/VMCB
- Guest physical memory
- Kernel devices – VHPET, VRTC, VLAPIC etc.
- Virtual devices – virtio devices, UART, AHCI

Save Mechanism



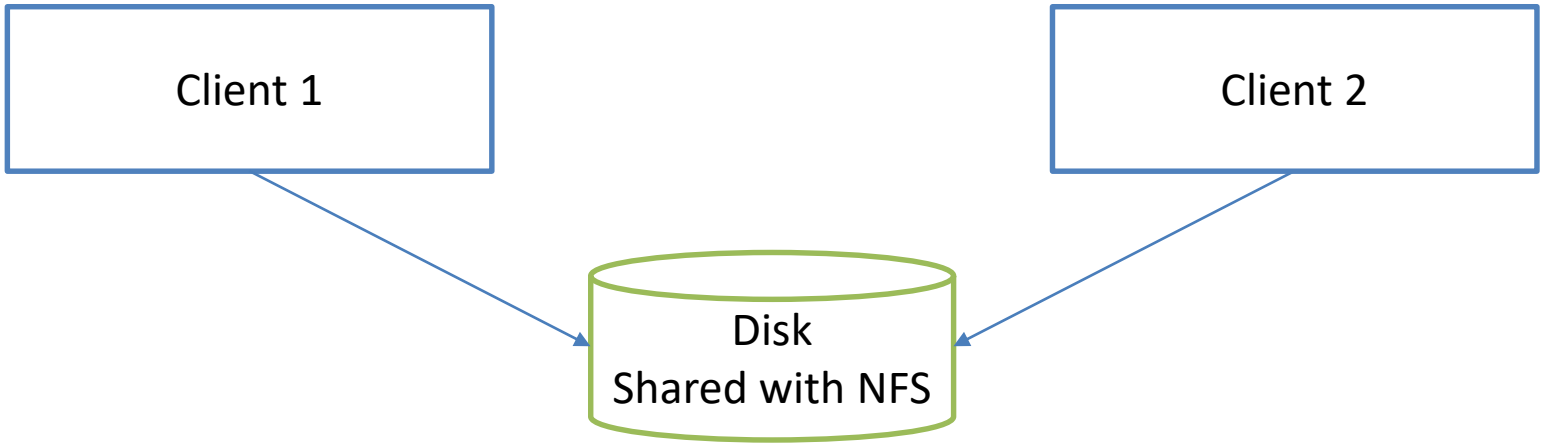
Restore Mechanism



Adding a migration feature for bhyve

- Based on the Save&Restore for bhyve Project
- Features to be presented:
  - Warm Migration for bhyve
  - Pre-Copy Live Migration approach for bhyve based on a Copy-on-Write Mechanism

Save-Restore “Migration”



1. Open VM
2. Snapshot VM
3. Close VM

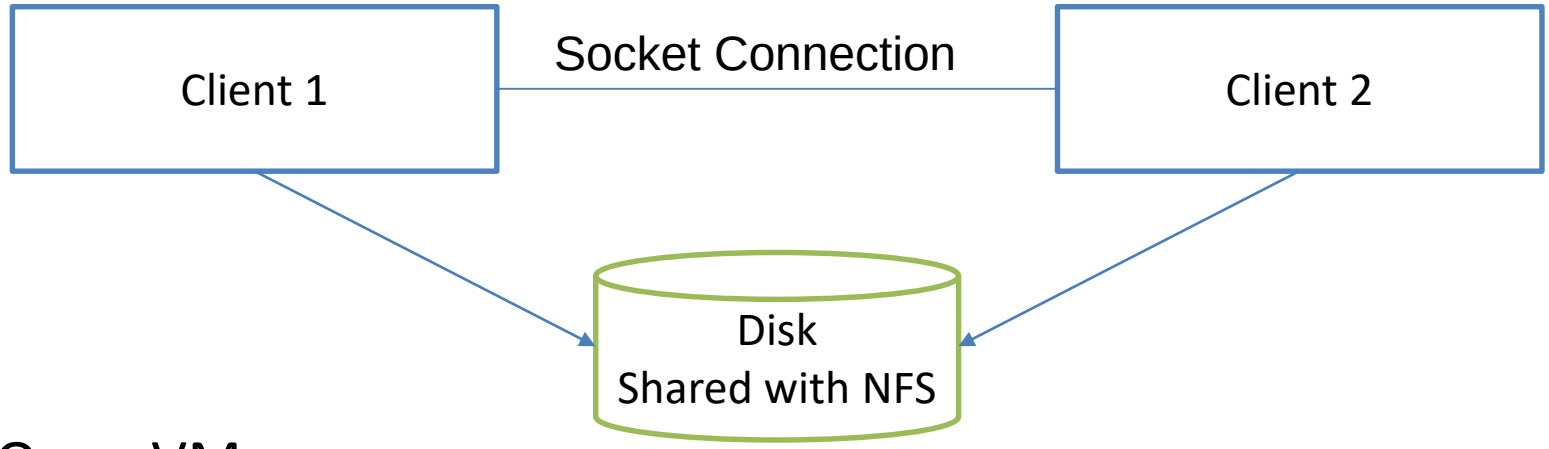
4. Restore VM

Save-Restore “Migration”

Limitations:

- User has to manually check if hosts are compatible for migration
- Additional space required for saving files
- Takes a lot of time

Warm Migration



1. Open VM

2. Open VM
3. Wait for Migration

4. Stop VM
5. Send state through socket

6. Receive state
7. Start VM

8. Destroy VM

Warm Migration - Usage

- Run VM

root@src # bhyve <options> vm_src

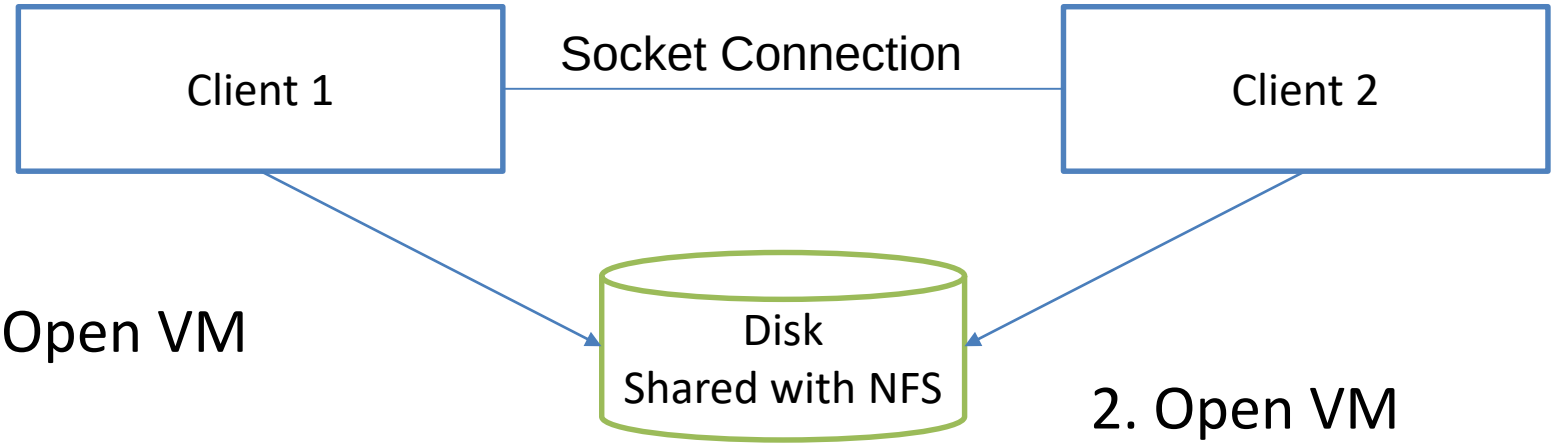
- Wait for migration

root@dst # bhyve <options> -R src_IP,port vm_dst

- Start Migration

root@src # bhyvectl --migrate=dst_IP,port vm_src

Live Migration



1. Open VM
2. Open VM
3. Wait for Migration
4. Send memory in rounds through socket
5. Receive memory
6. Stop VM
7. Send state
8. Receive state
9. Start VM
10. Destroy VM

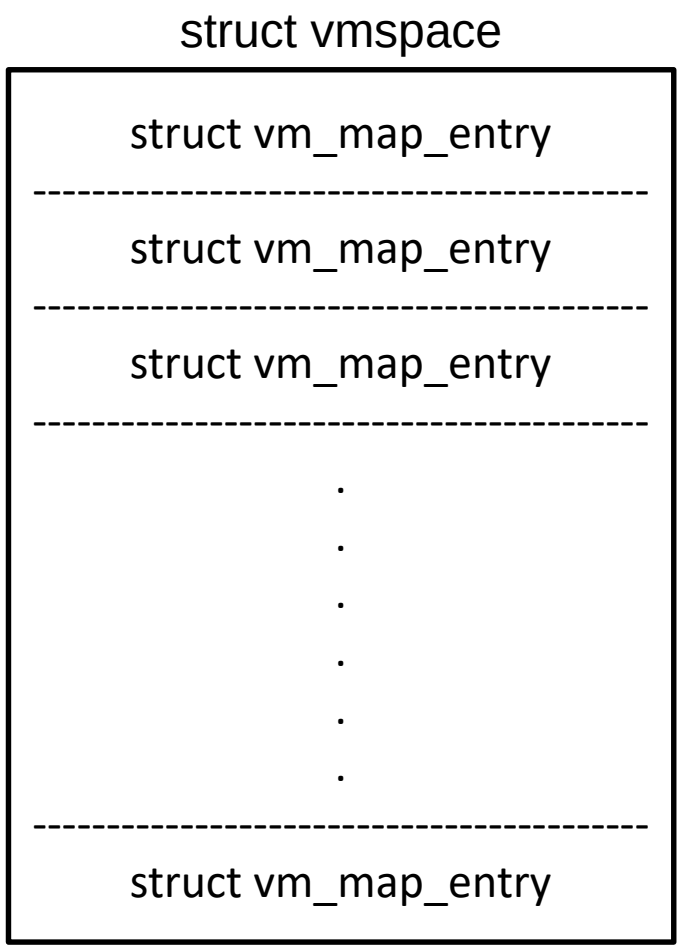
Live Migration Challenges

- The difficult part: live migrating the memory
- Memory is migrated in rounds
- Need to determine the memory pages that were modified since the last round started

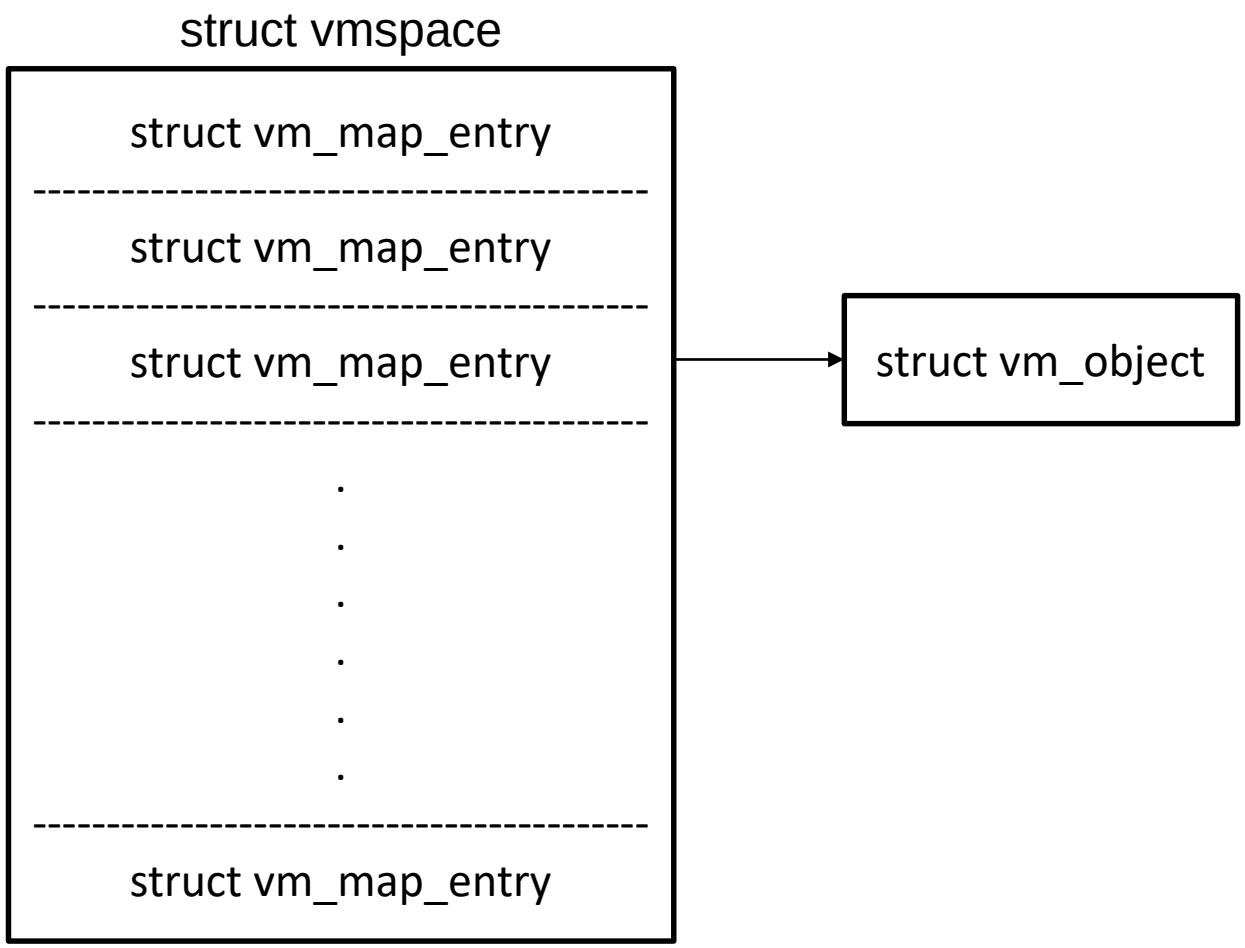
Live Migration using Copy-on-Write

- When spawning a process with fork(), its memory is marked as CoW
- Pages are duplicated when a write operation occurs
- Check the differences between the parent's memory and child's memory

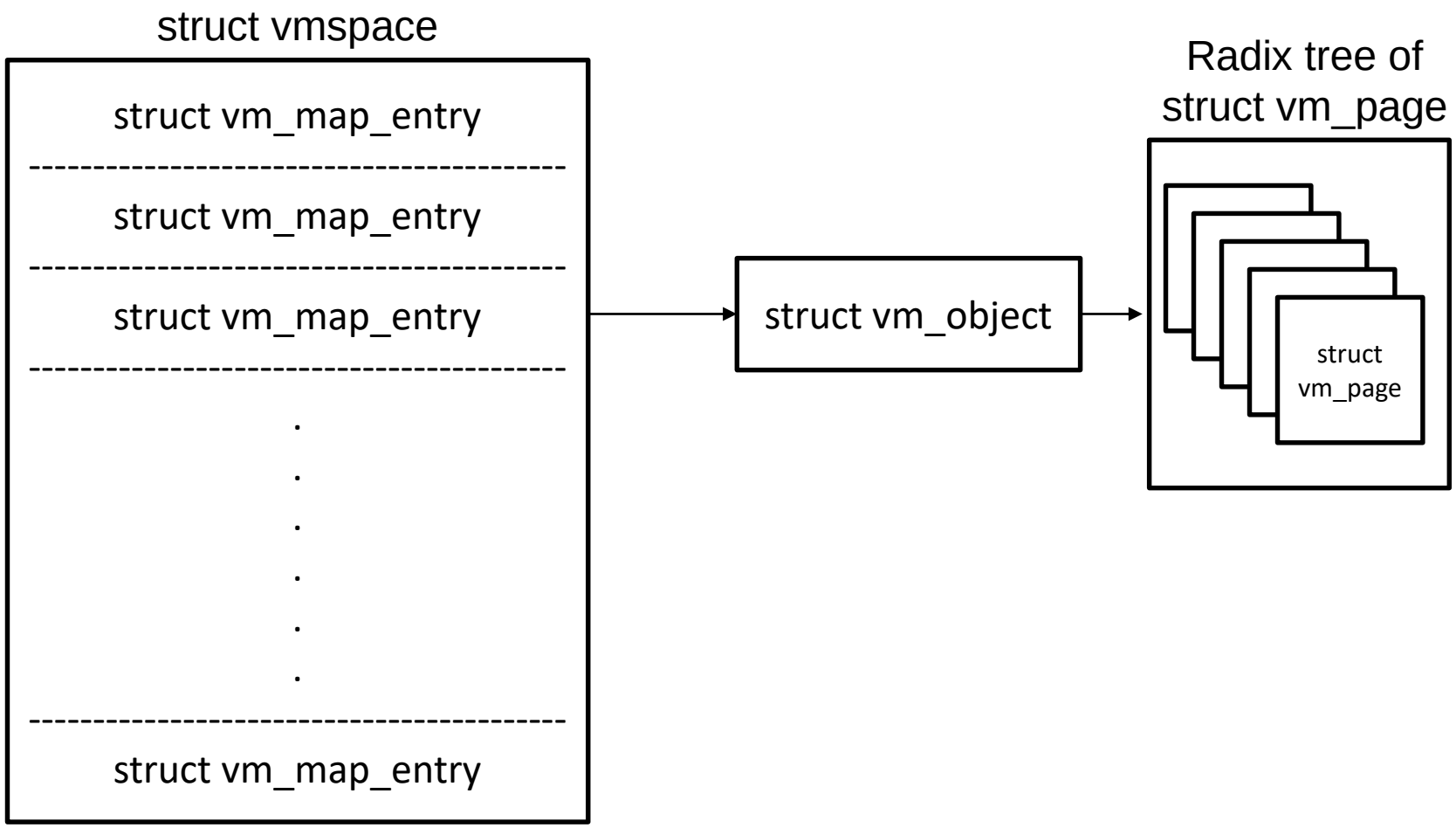
Virtual Memory Management in FreeBSD



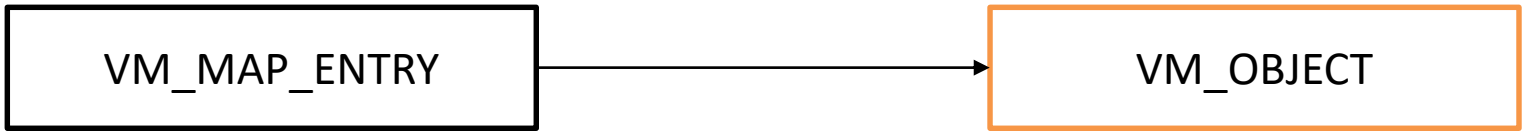
Virtual Memory Management in FreeBSD



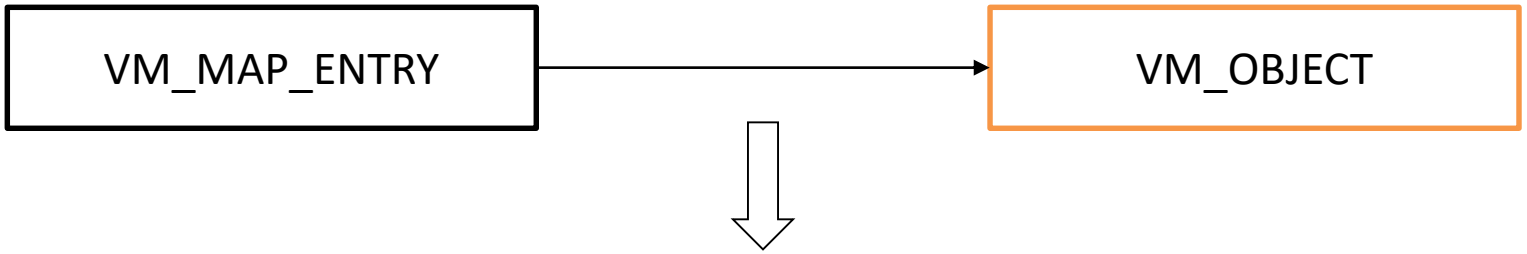
Virtual Memory Management in FreeBSD



Copy on Write in FreeBSD

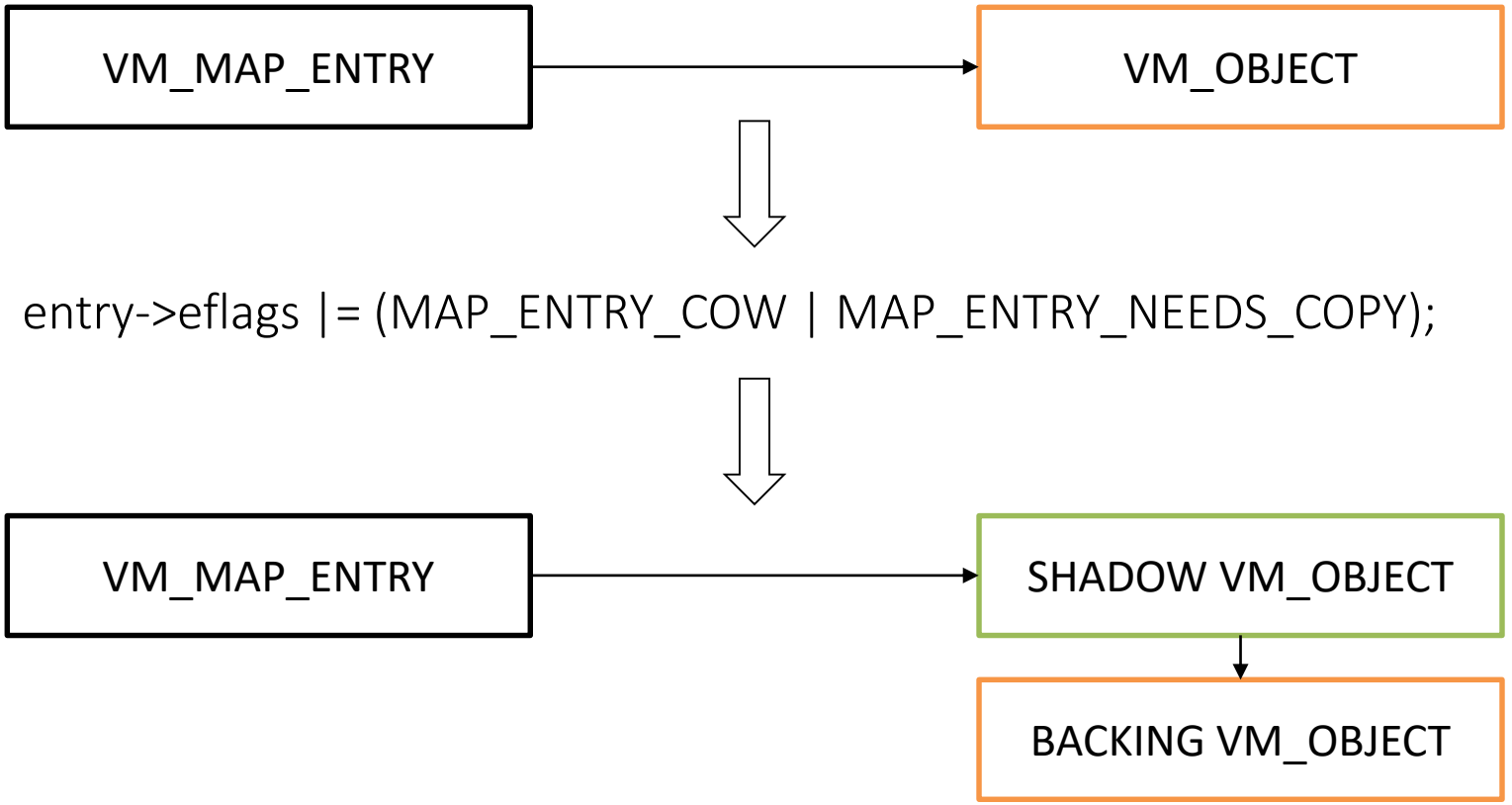


Copy on Write in FreeBSD

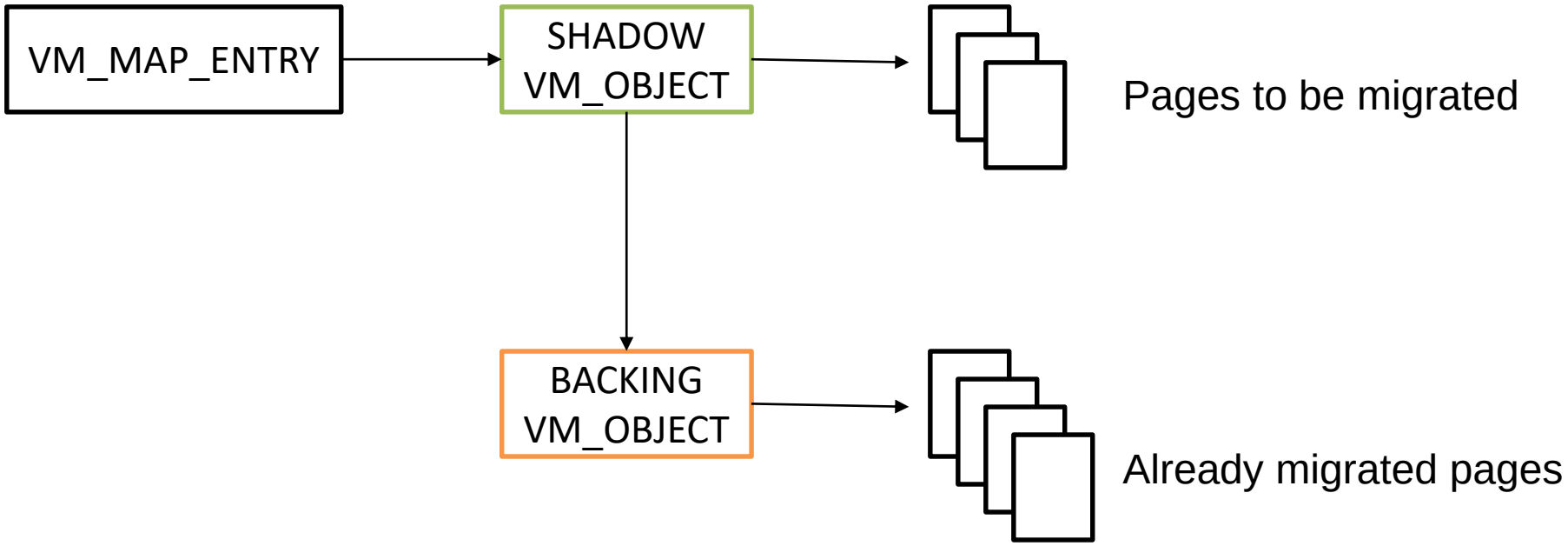



```
entry->eflags |= (MAP_ENTRY_COW | MAP_ENTRY_NEEDS_COPY);
```

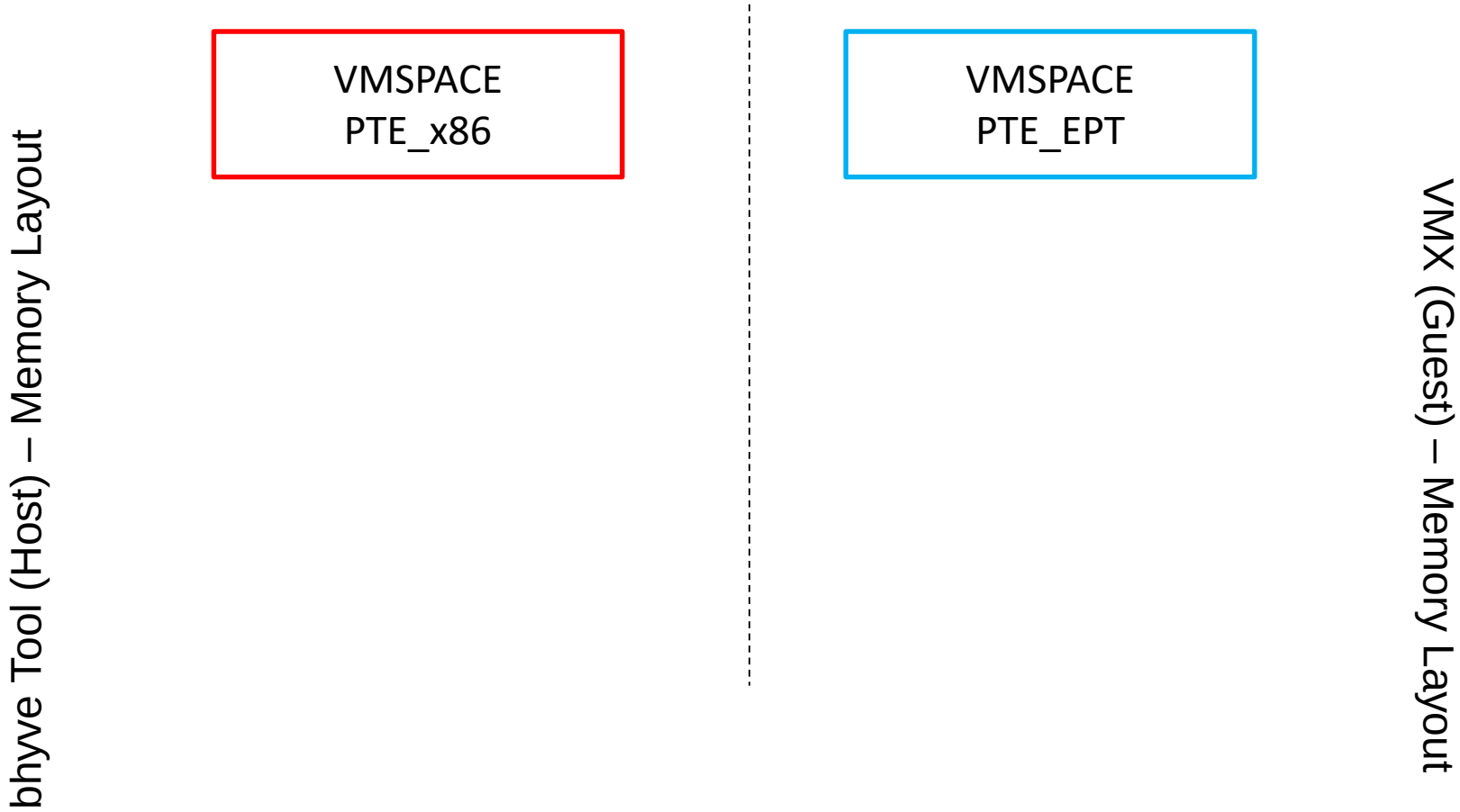
Copy on Write in FreeBSD



Copy-on-Write Guest Memory

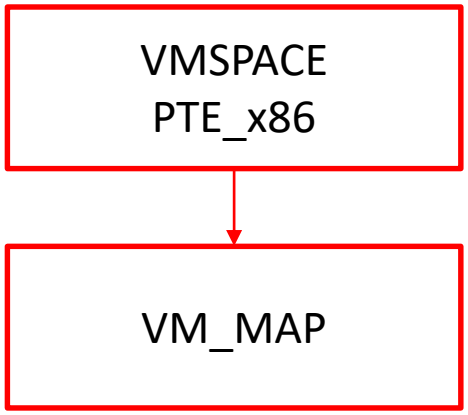


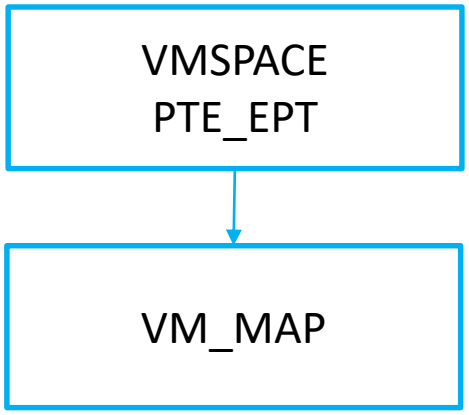
Bhyve – Memory Layout



Bhyve – Memory Layout

bhyve Tool (Host) – Memory Layout

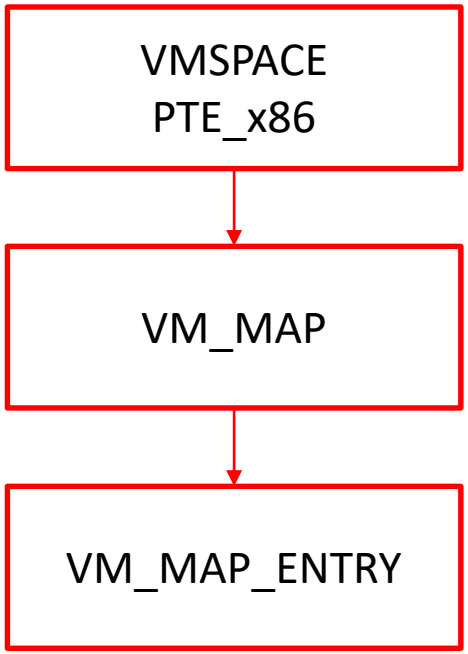


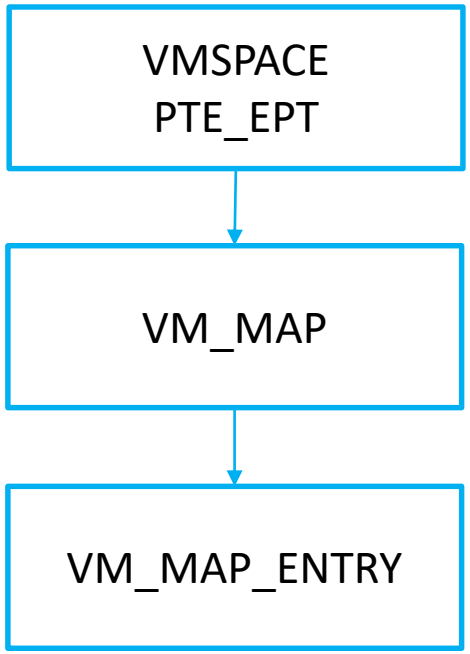


VMX (Guest) – Memory Layout

Bhyve – Memory Layout

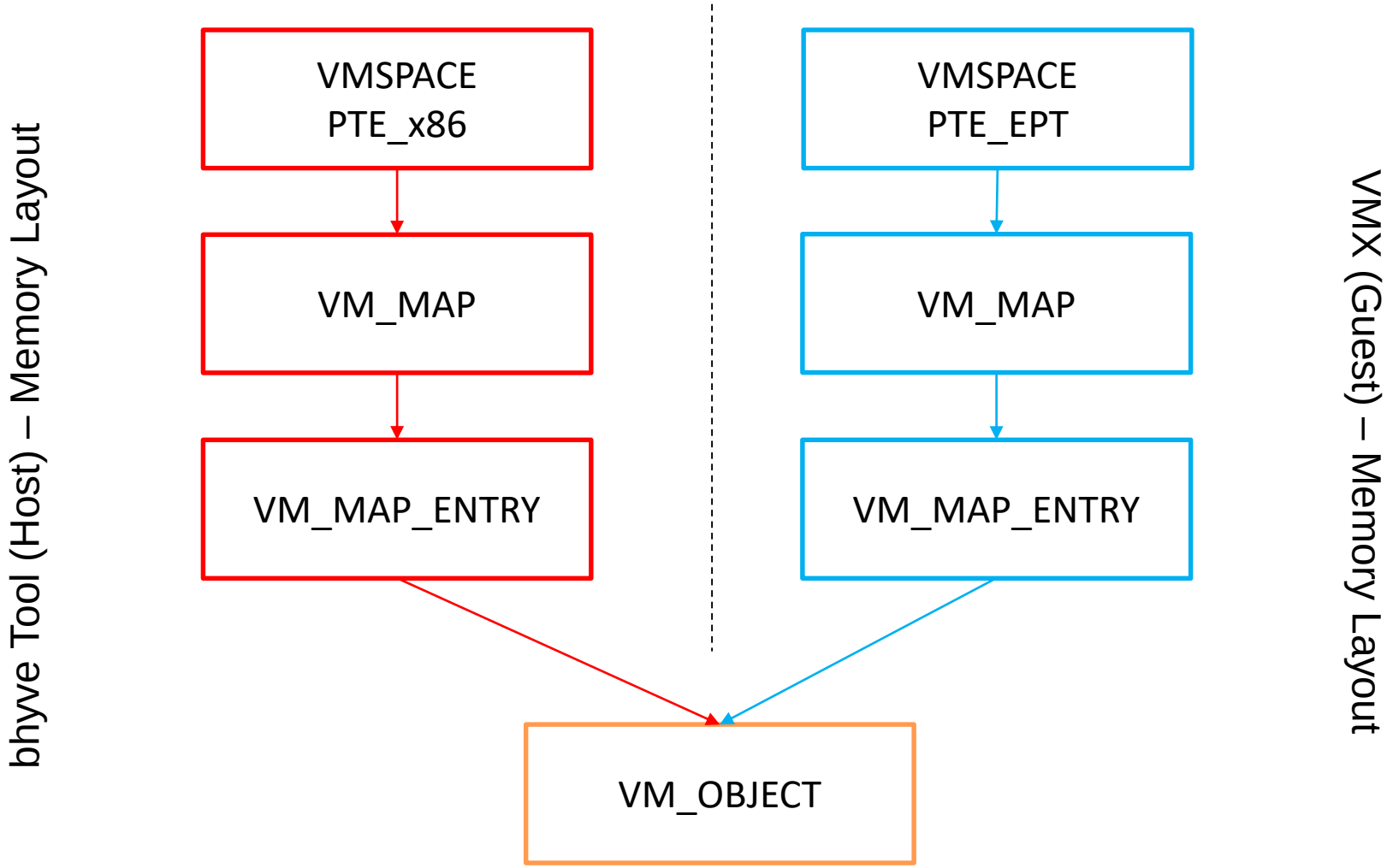
bhyve Tool (Host) – Memory Layout



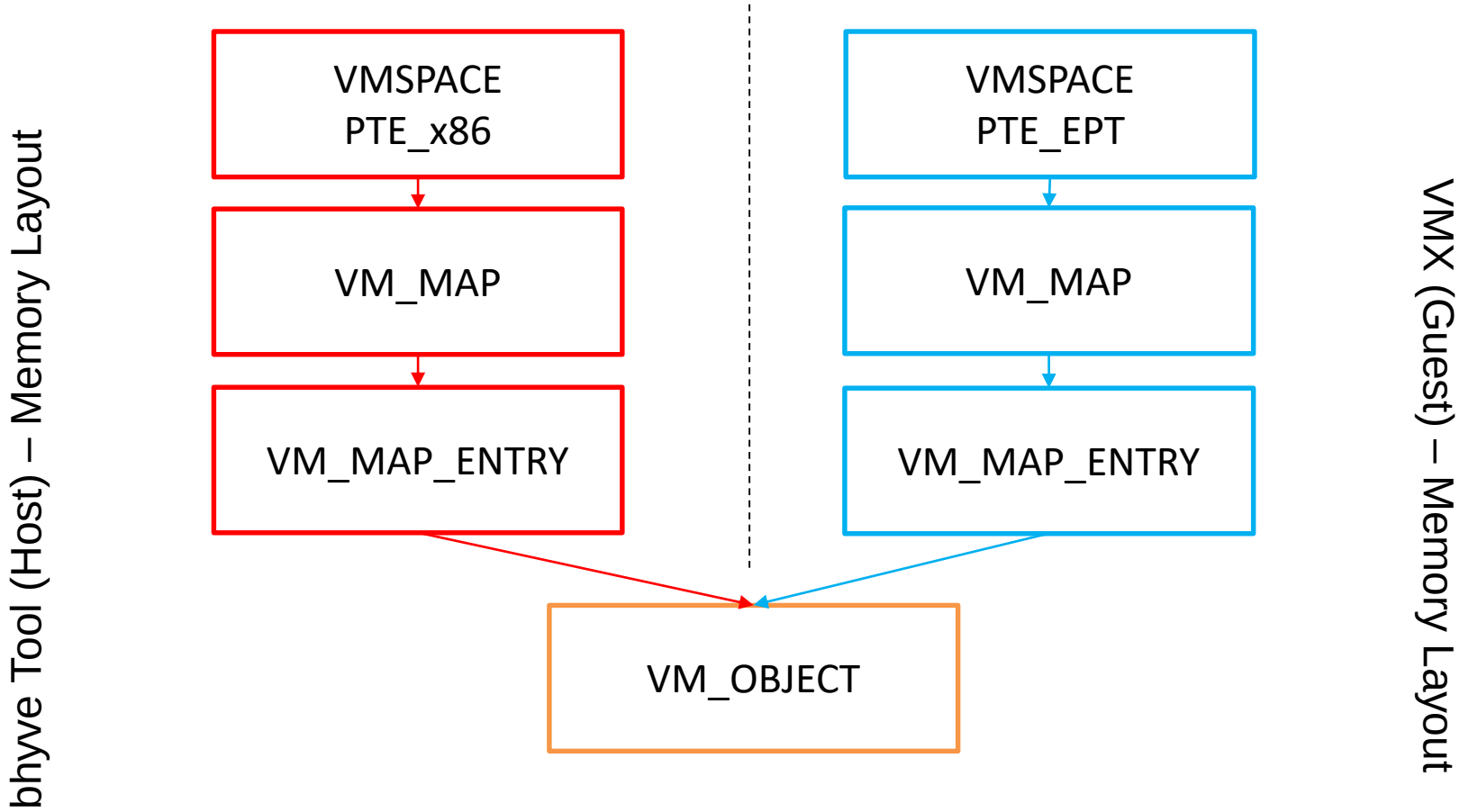


VMX (Guest) – Memory Layout

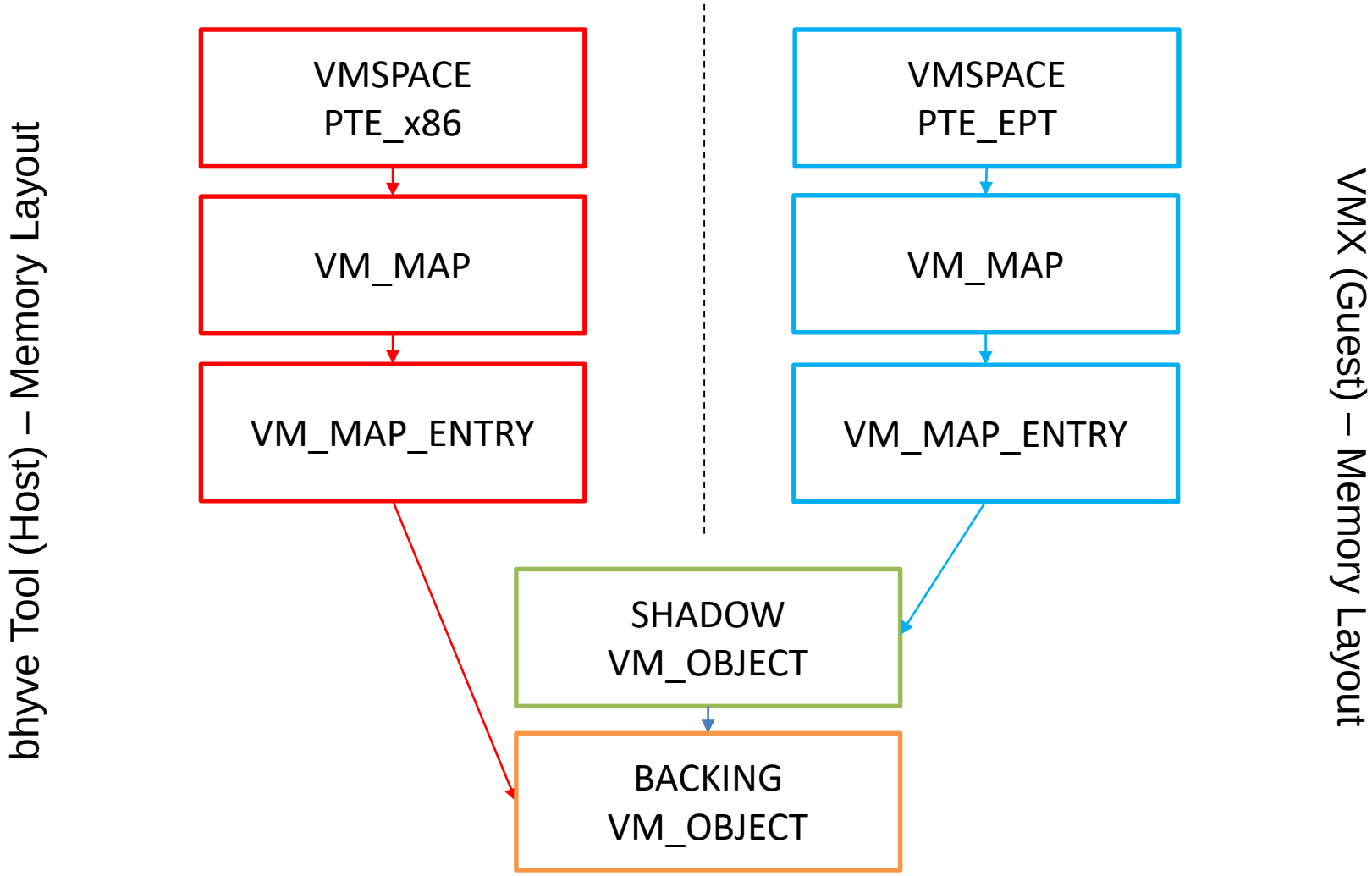
Bhyve – Memory Layout



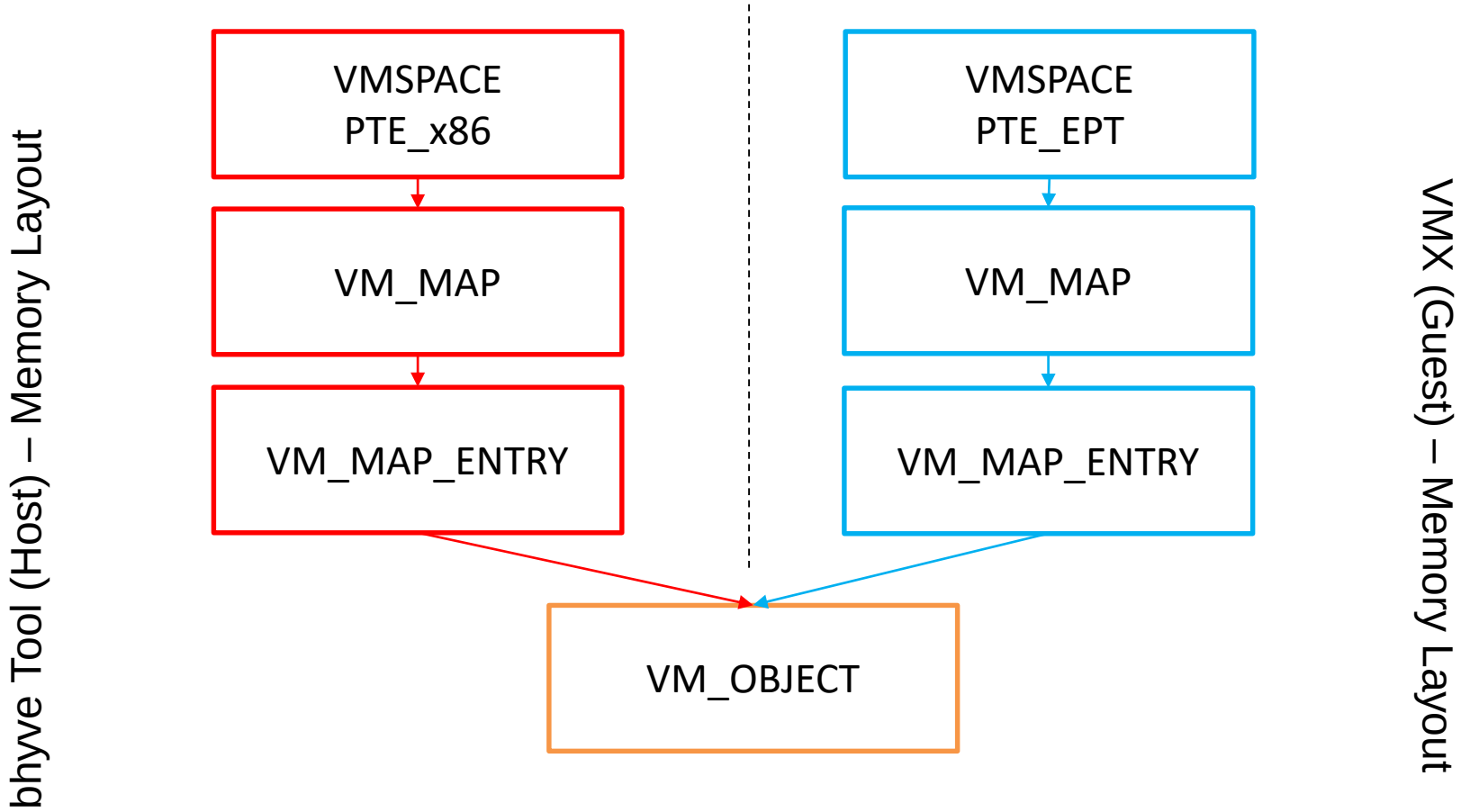
Copy-on-Write Guest Memory Object



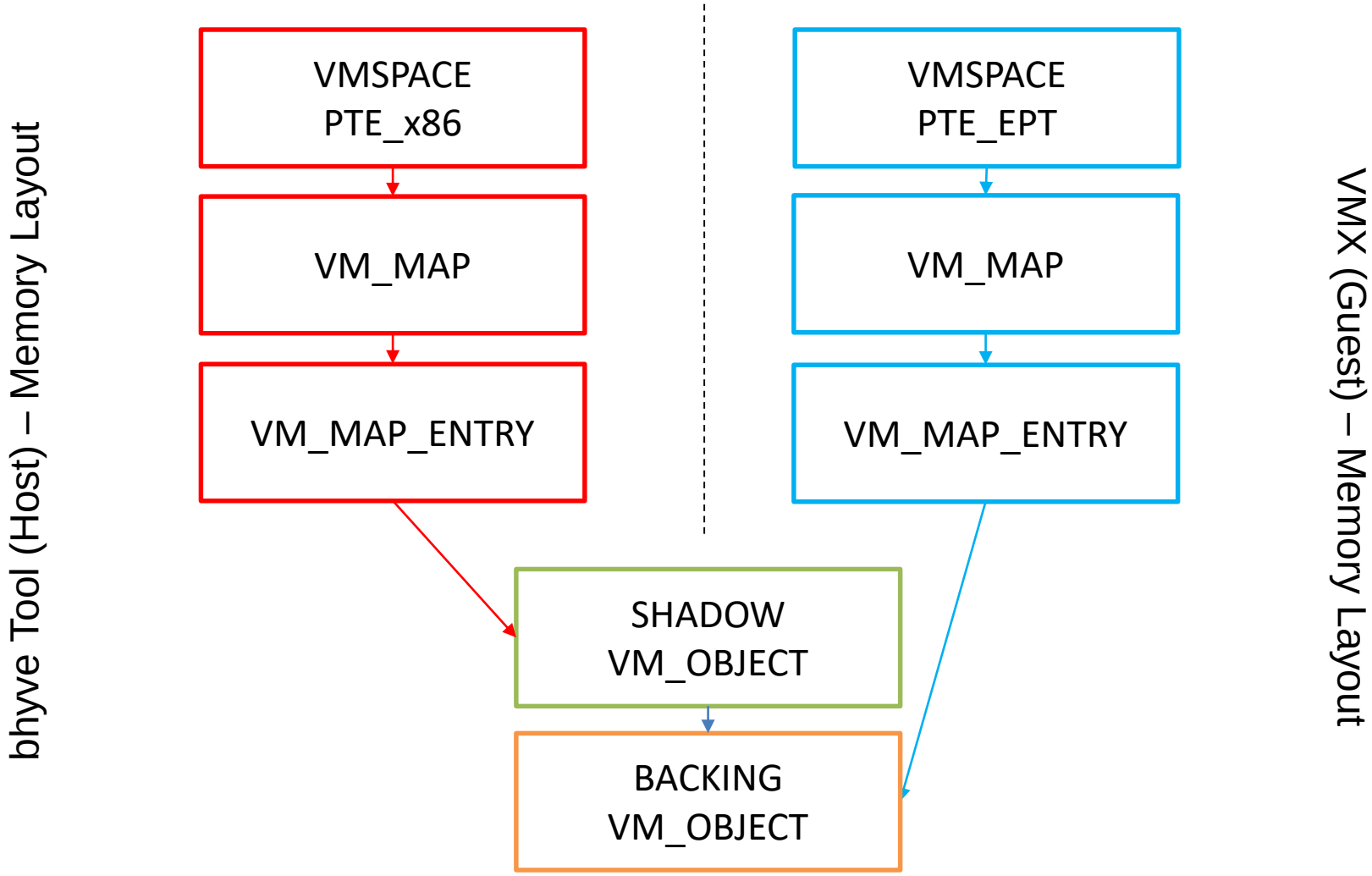
Copy-on-Write Guest Memory Object



Copy-on-Write bhyve Memory Object



Copy-on-Write bhyve Memory Object



Copy-on-Write Guest Memory

- Host and Guest won't see the same memory
- Communication between host and guest is lost (e.g., networking, block device access)
- Virtual Machine will eventually crash

New Approach

- We wanted to use Copy-on-Write to determine pages to be sent… but it doesn't work
- Next, dirty bits approach

Dirty bit approach

- use a dirty bit for each vm_page
- clear all the dirty bits at the beginning of a round
- in the next round, check all the dirtied pages and send them
- clear all the dirty bits and repeat the procedure

Dirty bit approach

- Each vm_page has a dirty flag field that is update from time to time based on the hardware Modified bit (AD bits)
- … but it cannot be used (vm_page's dirty flag is used by other subsystems; laundry systems)
- So we'll use our own dirty bit

Algorithm

1. Connect source and destination
2. Check for compatibility

   // First Migration Round
3. page_list = all guest's pages
4. send page_list to destination

Algorithm

5. for each remaining migration round – 1
6. page_list = []
7. search_for_dirty_pages(page_list)
8. send_to_dest(page_list)
9. end for

Algorithm

// Last Round

10. page_list = []
11. freeze_vm()
12. search_for_dirty_pages(page_list)
13. send_to_dest(page_list)
14. send_to_dest(kern_structs)
15. send_to_dest(devs)
16. send_to_dest(CPU state)

Implementation

- Use an unused bit from vm_page->oflags
- VPO_VMM_DIRTY

- Update VPO_VMM_DIRTY when vm_page->dirty is updated

- Clear VPO_VMM_DIRTY after a page is sent
- Force a sync

Implementation

- Iterate through all guest's vm_pages and retain indexes for the dirty ones
- Copy vm_pages into a userspace buffer and send it to destination via sockets and clean the dirty bit
- … and from the userspace buffer to vm_spaces (recv part)
- Add --migrate-live option in bhyvectl

Live Migration - Usage

- Run VM

root@src # bhyve <options> vm_src

- Wait for migration

root@dst # bhyve <options> -R src_IP,port vm_dst

- Start Migration

root@src # bhyvectl --migrate-live=dst_IP,port vm_src

Current Limitations

- Only with wired memory (otherwise pages can be swapped out)
- Number of rounds is static (4 in our case) – it should be chosen dynamically

Current Status and Future Work

What we have implemented

- Warm Migration and the framework for Live Migration

What we do now

- Improve Live Migration Support in bhyve

Special Thanks

- Mihai Carabaș, Darius Mihai, Sergiu Weisz
- Marcelo Araujo

- John Baldwin, Mark Johnston, Alan Cox

- Matthew Grooms for financial support

FreeBSD-UPB on Github

- Save-Restore Project:
  - https://github.com/FreeBSD-UPB/freebsd/tree/projects/bhyve_snapshot
- Warm Migration Project:
  - https://github.com/FreeBSD-UPB/freebsd/tree/projects/bhyve_warm_migration
- Live Migration Project:
  - https://github.com/FreeBSD-UPB/freebsd/tree/projects/bhyve_migration_dev

FreeBSD-UPB on Github

- Save/Restore – How To Use:
  - https://github.com/FreeBSD-UPB/freebsd/wiki/Save-and-Restore-a-virtual-machine-using-bhyve
- Warm Migration and Live Migration – How to Use:
  - https://github.com/FreeBSD-UPB/freebsd/wiki/Virtual-Machine-Migration-using-bhyve

References

- Pictures and Logos:
  - http://bhyve.org/static/bhyve.png
  - https://www.freebsdfoundation.org/about/project/
- FreeBSD Virtual Memory Management:
  - *Design elements of the FreeBSD VM system*, Matthew Dillon [Online] https://www.freebsd.org/doc/en/articles/vm-design/ [Accessed Dec, 21st, 2018]
  - https://github.com/freebsd/freebsd
- Bhyve Memory Layout:
  - *Nested Paging in bhyve*, N. Natu, P. Grehan
  - https://github.com/freebsd/freebsd

References

- Migration Documentation:
  - *Virtual Machine Migration* [Online] https://nsrc.org/workshops/2014/sanog23-virtualization/raw-attachment/wiki/Agenda/migration-storage.pdf [Accessed Dec, 21st, 2018].
  - *VMware virtual machine migration types vSphere 6.0*, VMWare [Online] https://communities.vmware.com/docs/DOC-31922 [Accessed Dec, 21st, 2018]

References

- Migration Documentation:
  - *Live migration of virtual machines*, C. Clark, and K. Fraser and S. Hand, and J.G. Hansen, and E. Jul, and C. Limpach, and I. Pratt, and A. Warfield.
  - *Post-copy live migration of virtual machines*, M.R. Hines and U. Deshpande and K. Gopalan
  - *kvm: the Linux virtual machine monitor*, A. Kivity and Y. Kamay and D. Laor and U. Lublin and A. Liguori