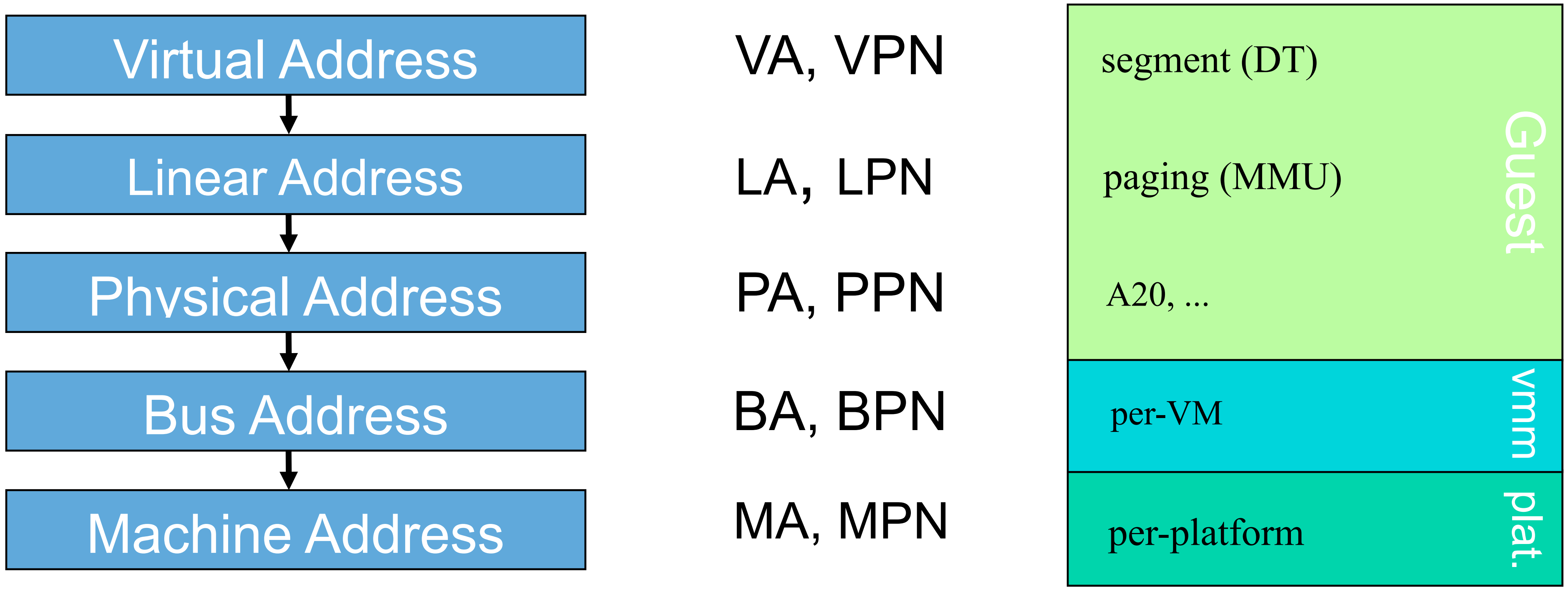
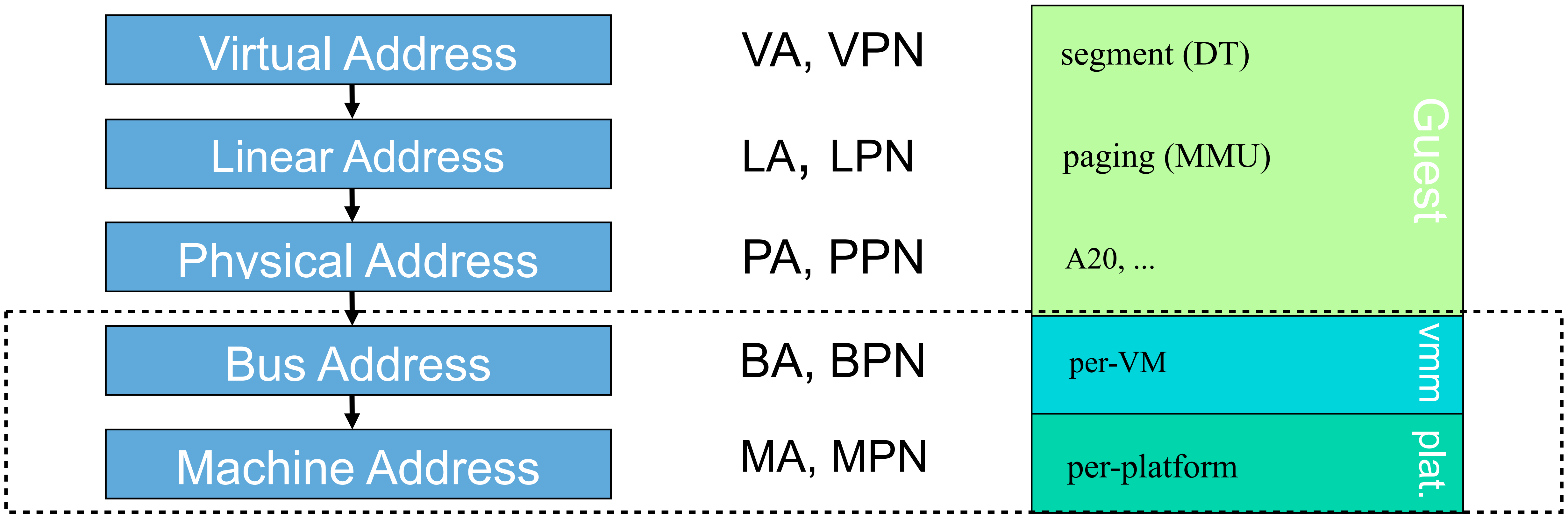
BusMem and Memory Scheduling

Alex Garthwaite
MonitorU
July 27th, 2010

Background

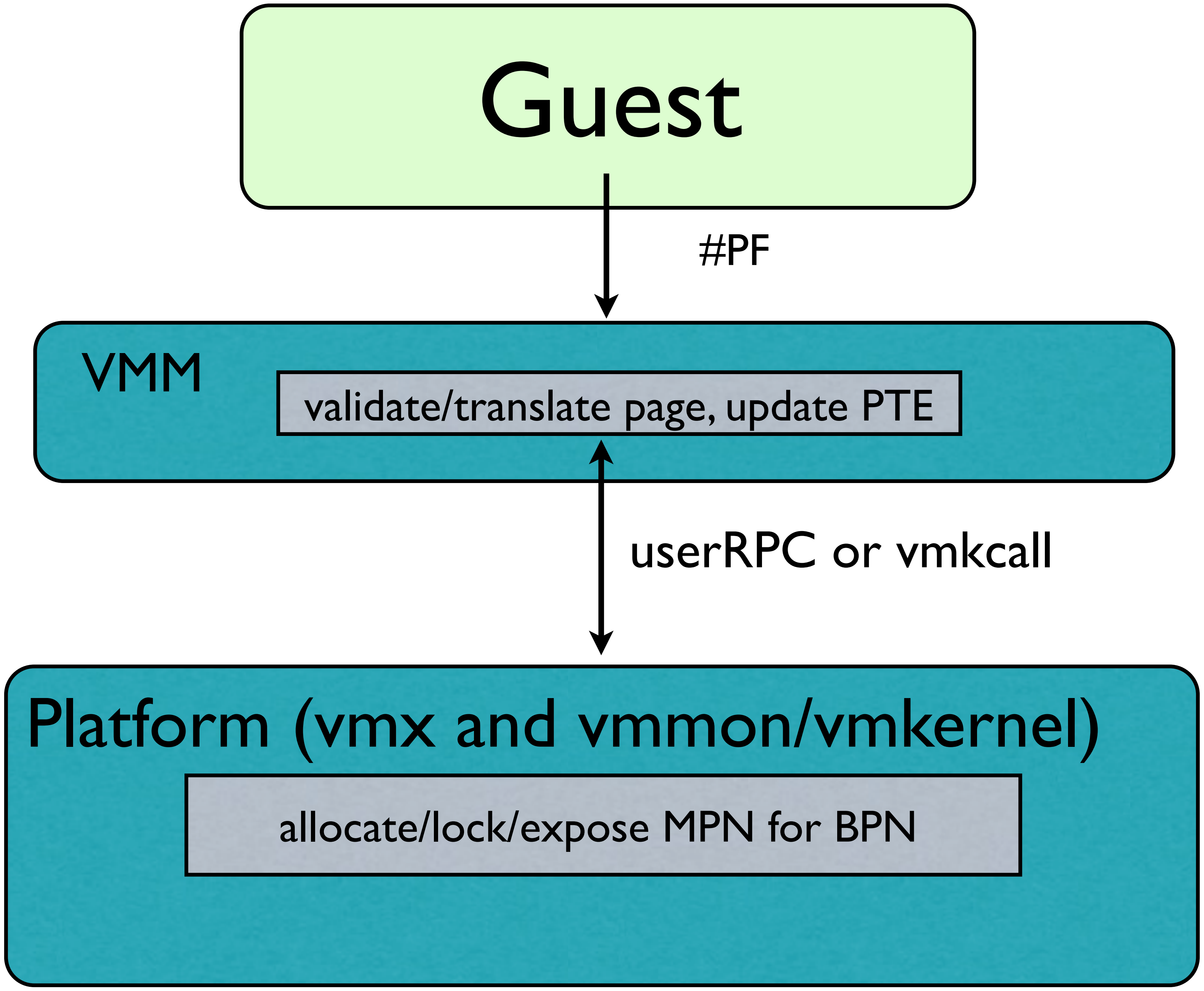

Virtual Address
Linear Address
Physical Address
Bus Address
Machine Address
VA, VPN
LA, LPN
PA, PPN
BA, BPN
MA, MPN
segment (DT)
paging (MMU)
A20, ...
Guest
per-VM
vmm
per-platform
plat.

Background



BusMem: map BPNs to MPNs and track state

Guest Memory



Guest
#PF
VMM
validate/translate page, update PTE
userRPC or vmkcall
Platform (vmx and vmmon/vmkernel)
allocate/lock/expose MPN for BPN

Topics

- Memory state in the vmm
- Translation/zapping/invalidation
- Memory scheduling and services
- Some new directions

Not Covered

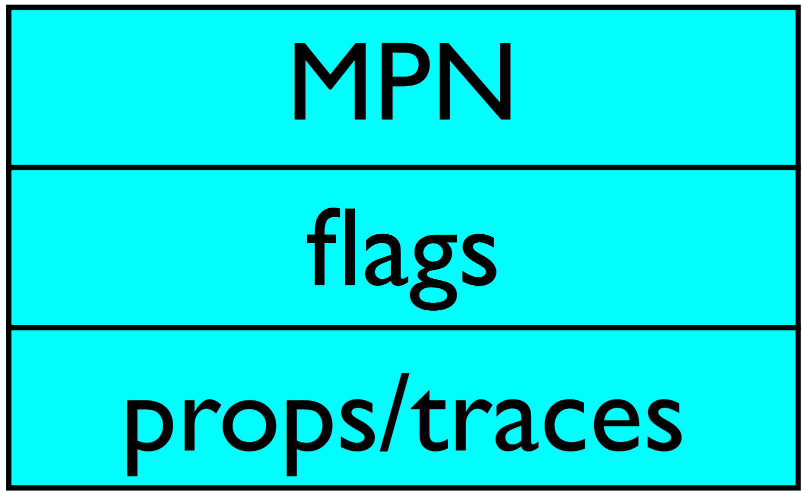
- VM states and allocation
  - power-on/off
  - checkpointing (of memory)/migration
- How traces on memory are used
- Overheadmem and VM admission
- Monitor actions
- Details of large/small page management
- Anon page allocation/release
- ...

Themes

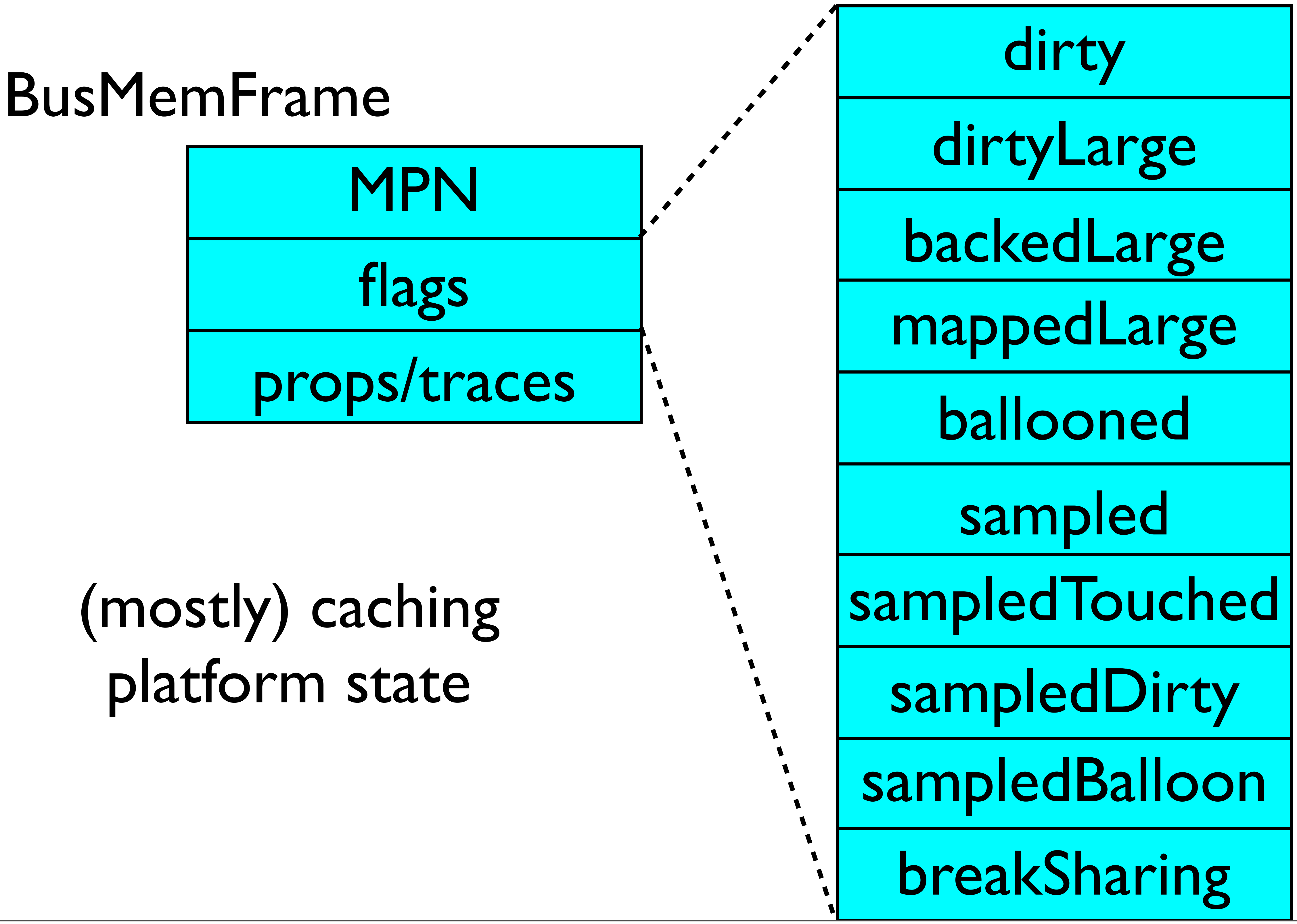
- Scaling and BusMem lock contention
- Overheads from tracking BPN state
- Release of memory and monitor actions
  - swapping/page-sharing/breaking sharing
- Coordination of policy/mechanisms across layers in and across VMs and platforms

BusMem Frames

BusMemFrame

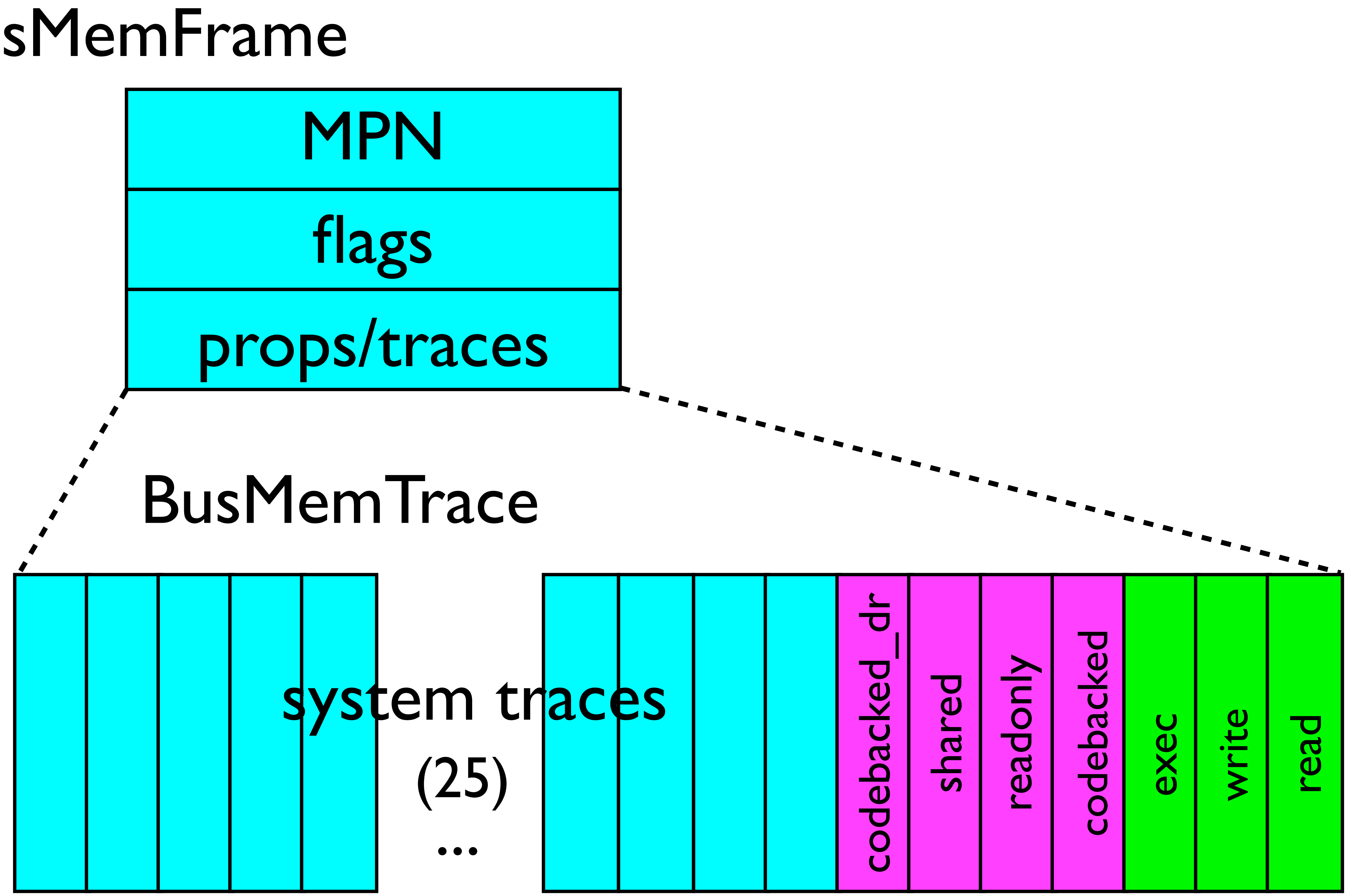


BusMem Frames

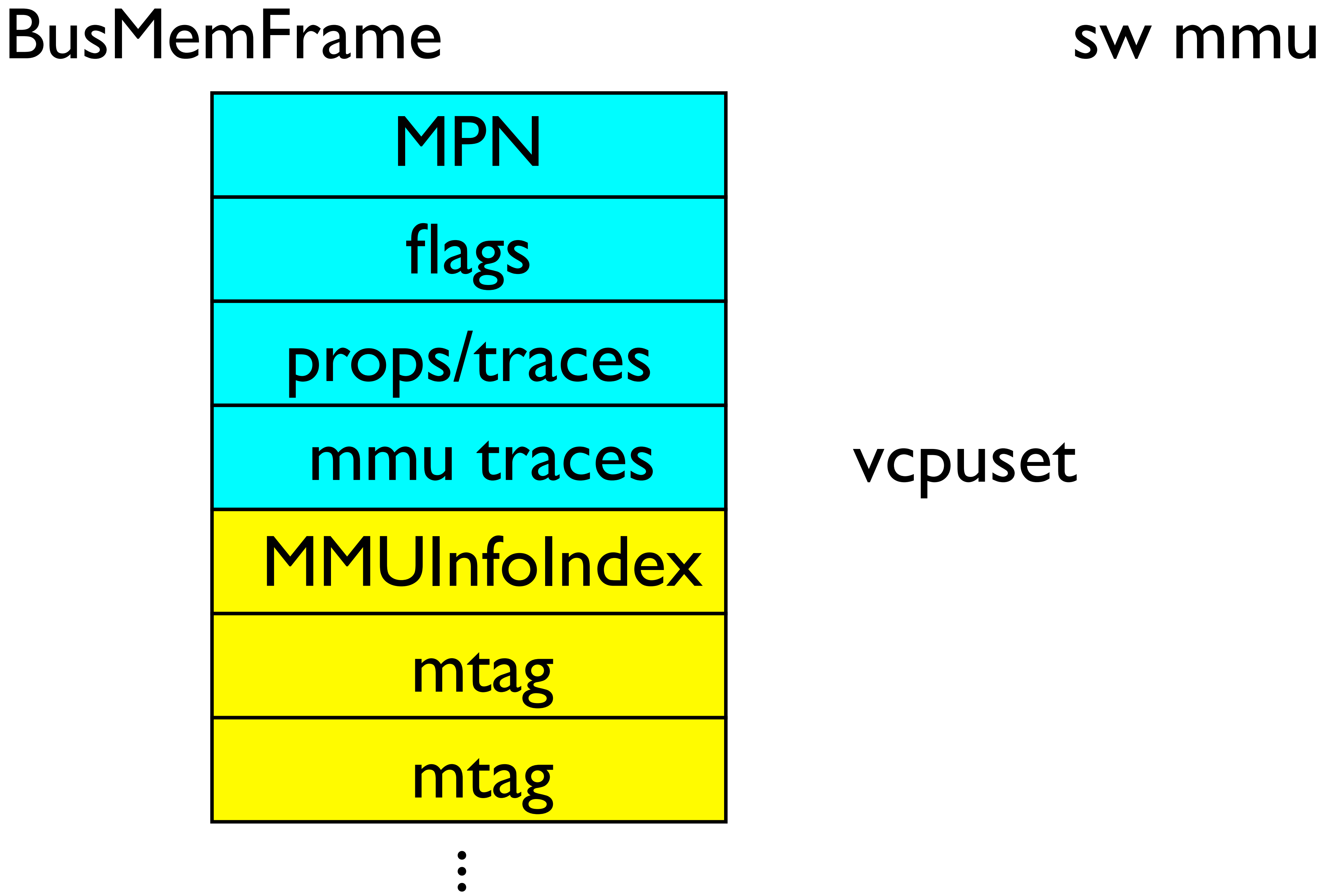


(mostly) caching
platform state

BusMem Frames



BusMem Frames



Translation/Validation

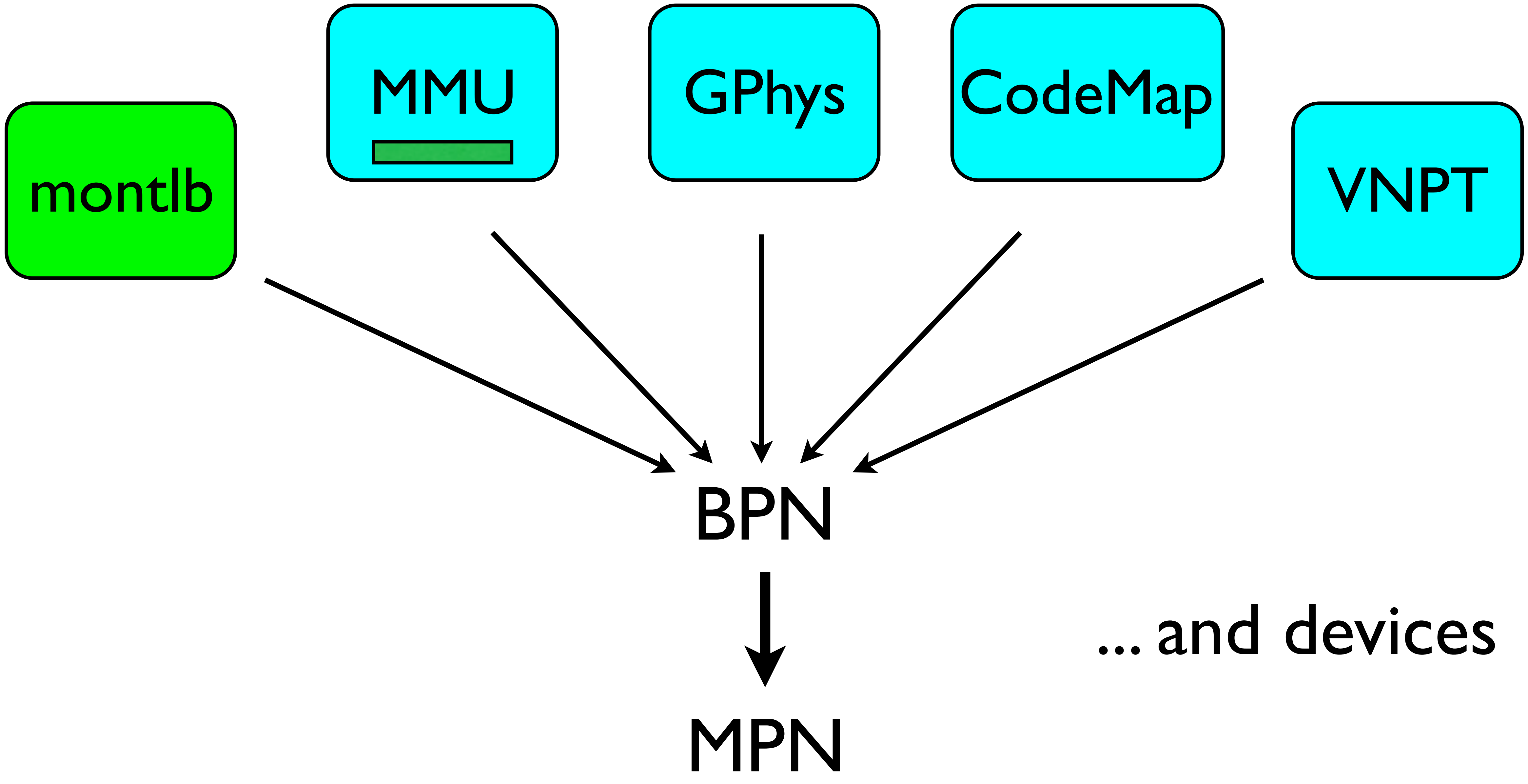
- Translation: BPN => MPN from BusMem
- Validation: call platform for BPN => MPN

```
read MPN
if (slow-path check) {
    do {
        open transaction if MPN invalid, breakcow
        Platform_LockGuestPage to get MPN if
            invalid, breakcow, or needs dirtying
        BusMemLock();
        validate MPN/state of BPN
        BusMemUnlock();
    } while (MPN not valid);
}
pin BPN
```

Translation/Validation

- Slow-path check
  - INVALID_MPN, breakcow and shared, sampling, needs dirtying
- Breaking sharing
  - frame->breakSharing = 1
  - take BusMemLock, invalidate, clear MPN/COW, release BusMemLock
- Validation checks

BusMem Clients



... and devices

Locking/Pinning

- Locking: fix mapping (BPN => MPN)
- Pinning: prevent release of mapping
  - examples: ESX p2mCache, vmx PhysMem, BusMem

PinCounts (shared)

BPN	count
...	...

PinSlots (per-vcpu)

BPN	
	(montlb)
...	
	(mmu)

Invalidation/Zapping

- Invalidation: request zap, allow for MPN to change or be released
- Zapping: notify clients to drop cached information about BPNs
- Protocol to support translation fast-path
  - MPN <= INVALID_MPN before crosscall
- Coallescing concurrent invalidations and crosscall/lock contention

Invalidation/Zapping

- Five kinds of invalidation
  - PageList from {swapping, page-sharing, p2mUpdate, remap}
  - range of BPNs from {large pages, p2mUpdate, region-invalidation}
  - single BPN from {balloon, breakcow}
  - sampling
  - physMem map/unmap

Invalidation/Zapping

- Steps:
  - acquire BusMem lock
  - mark BPNs' frames (INVALID_MPN)
  - check for pinned pages/mark cases
  - zap unpinned pages on all vcpus
  - clean up BPNs' frames
  - release BusMem lock

MemSched Services

release

- page-sharing
- ballooning
- swapping

reallocation

- remapping
- p2mUpdate
- VMCI

usage

- sampling
- checking

Page-sharing

- 100ms periodic task
  - randomly sample, hash, share collisions
  - platform match hints
- Cost of breaking sharing
  - p2mUpdate mechanism
- Invariants:
  - stop so frameMPN usable
  - rate-limit to minimize impact on vcpu
- Java ballooning and guest-hints

Ballooning

guest knowledge

asynchronous/can be slow

reserves PPNs

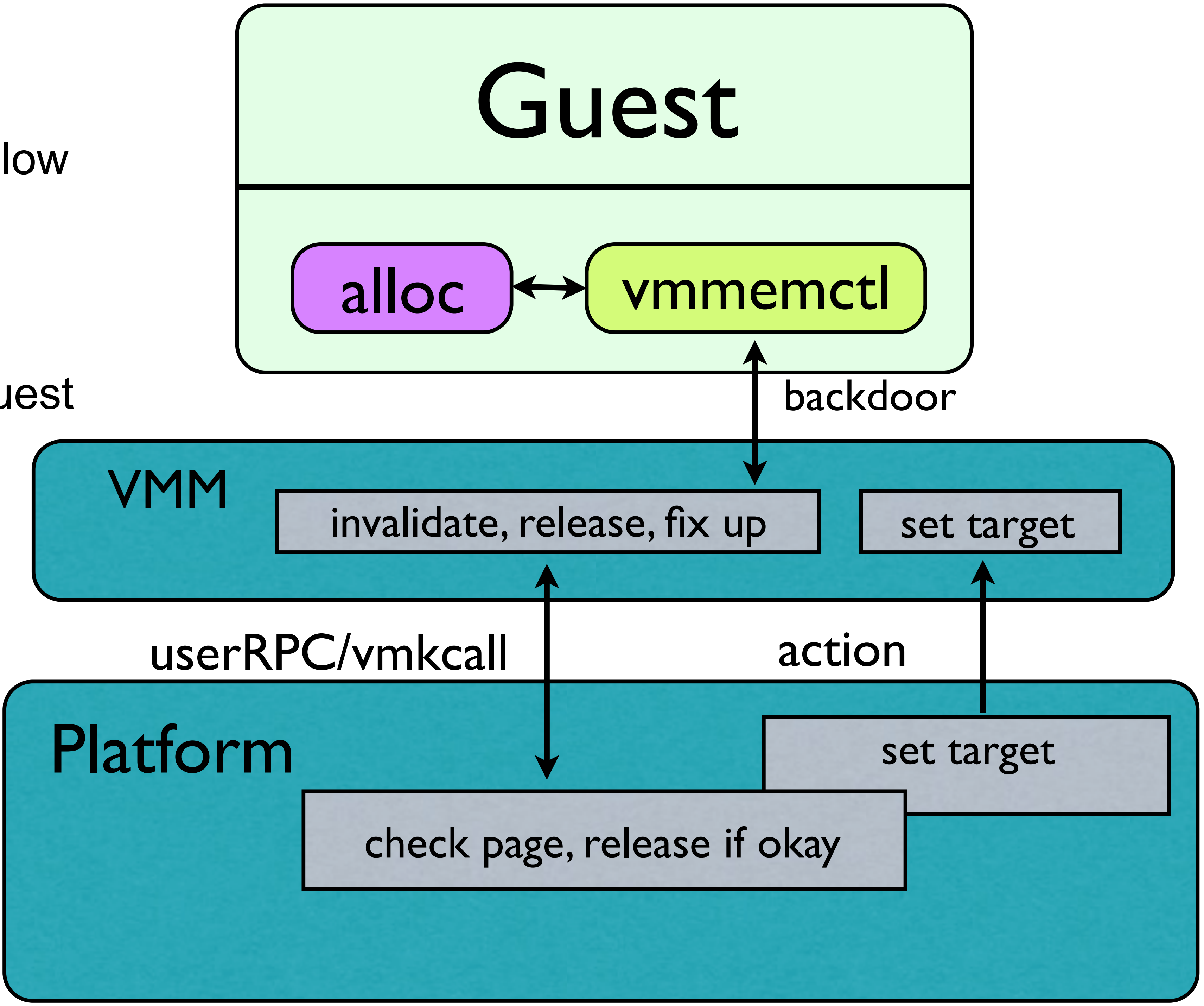
PPNs determined by guest

may not reclaim MPNs

may be reset by guest

relatively inflexible

shared zeroPage



Swapping

random choice

synchronous/transparent

one userRPC per swap-set

guest touches page in memory

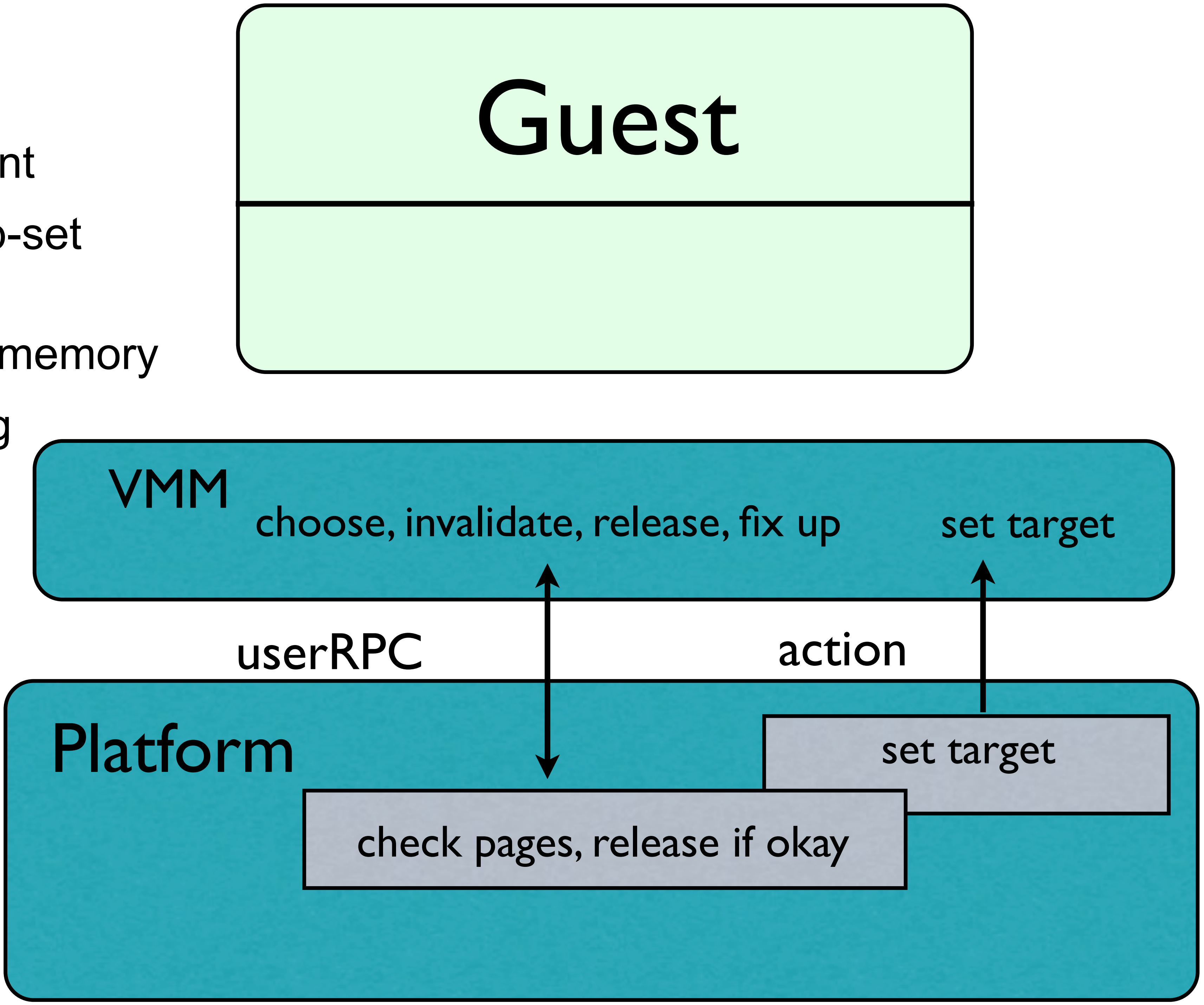
interaction w/ballooning

MN changes:

selection in platform

swap/allocation rates

avoid stopping vcpus


Guest
VMM
choose, invalidate, release, fix up
set target
userRPC
action
Platform
set target
check pages, release if okay

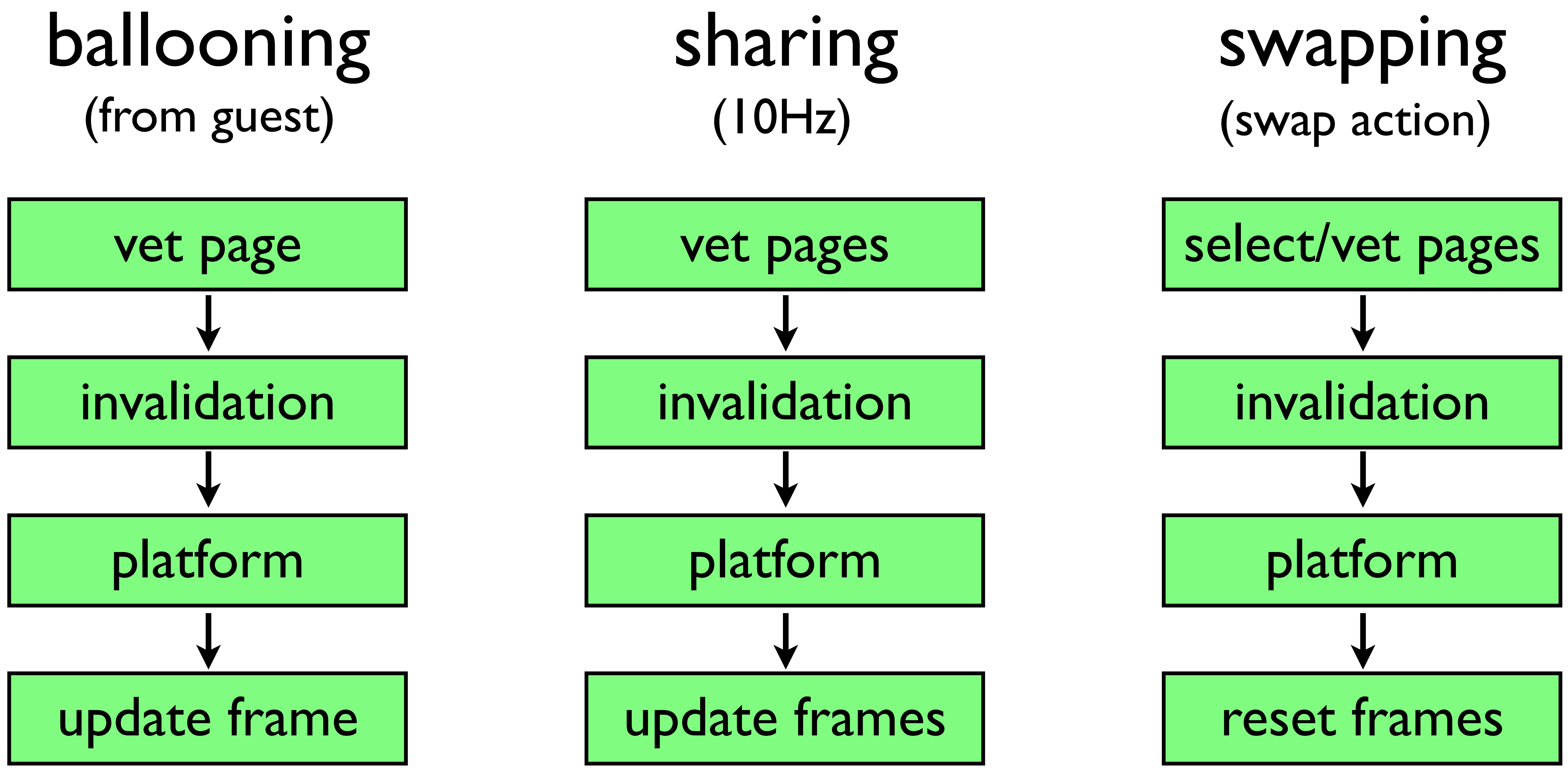
Remapping

- Use cases:
  - NUMA-migration
  - large-page defragmentation
  - page-retirement
  - page-coloring
- Get PageList batch, invalidate, call platform for new MPNs, update BPNs' frames

Sampling

- Four sets of 100 pages each, randomly
- Once per minute, oldest set killed, new one created
- Tracks avg. working-set size
  - used to balance memory across VMs
- Zaps without checking pinned state

MemSched Services

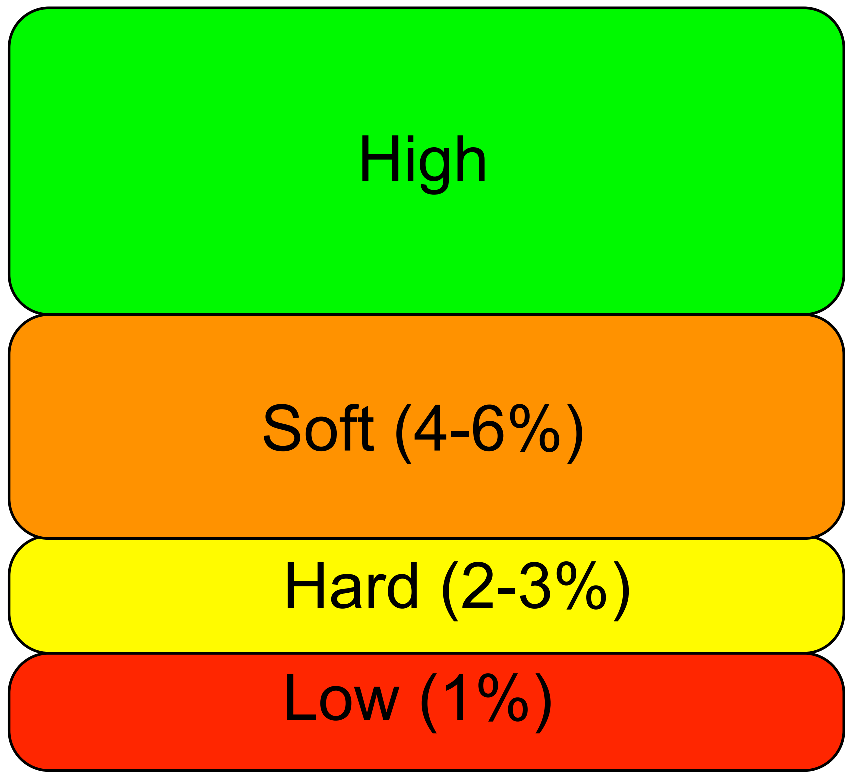


MemSched

Cooperative model with platform scheduler

- declarative model (currently)
- balance memory across VMs
  - based on memory usage (active wss, idle-page tax)
  - approx. lottery scheduling
  - once per second

memschedInfo in sharedArea and actions



Low Memory State

- Correctness issue
- Monitor actions and repeated swapping
- Should be harder to enter low state
  - allocation and swap rates
  - avoid need to stop vcpus
  - work (in RMX): userworld swapping
  - not-exposed-to-vmm

New Directions

- Distinguish translated from valid BPNs
  - improve locking/concurrency
  - eliminate stopping, avoid crosscalls
- Bulk-validation/invalidation
  - source of info. from guest and platform
- Restructure services
  - common structure/mechanism in vmm
  - tie reclamation/allocation rates
  - async page-sharing
  - selection, not-exposed pools
- New sampling techniques (miss-ratio curves)

Who's Involved?

Platform infra.

monitor group

ESX resource mgmt (rmx)

RM/monitor performance

vm-working-set@vmware.com

Mondays, 11am-12:30pm PDT, C13

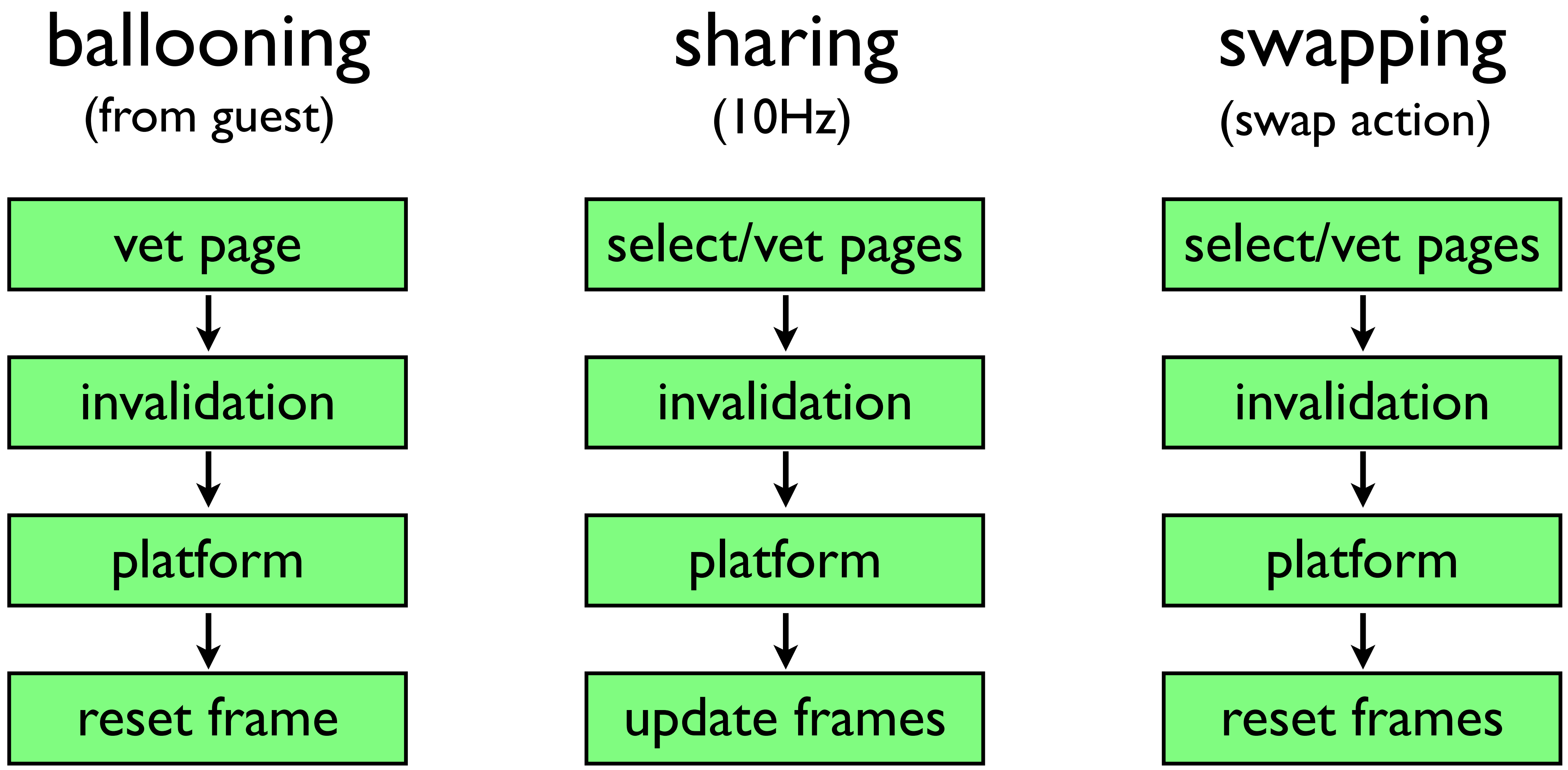
Questions?

https://wiki.eng.vmware.com/MonitorU

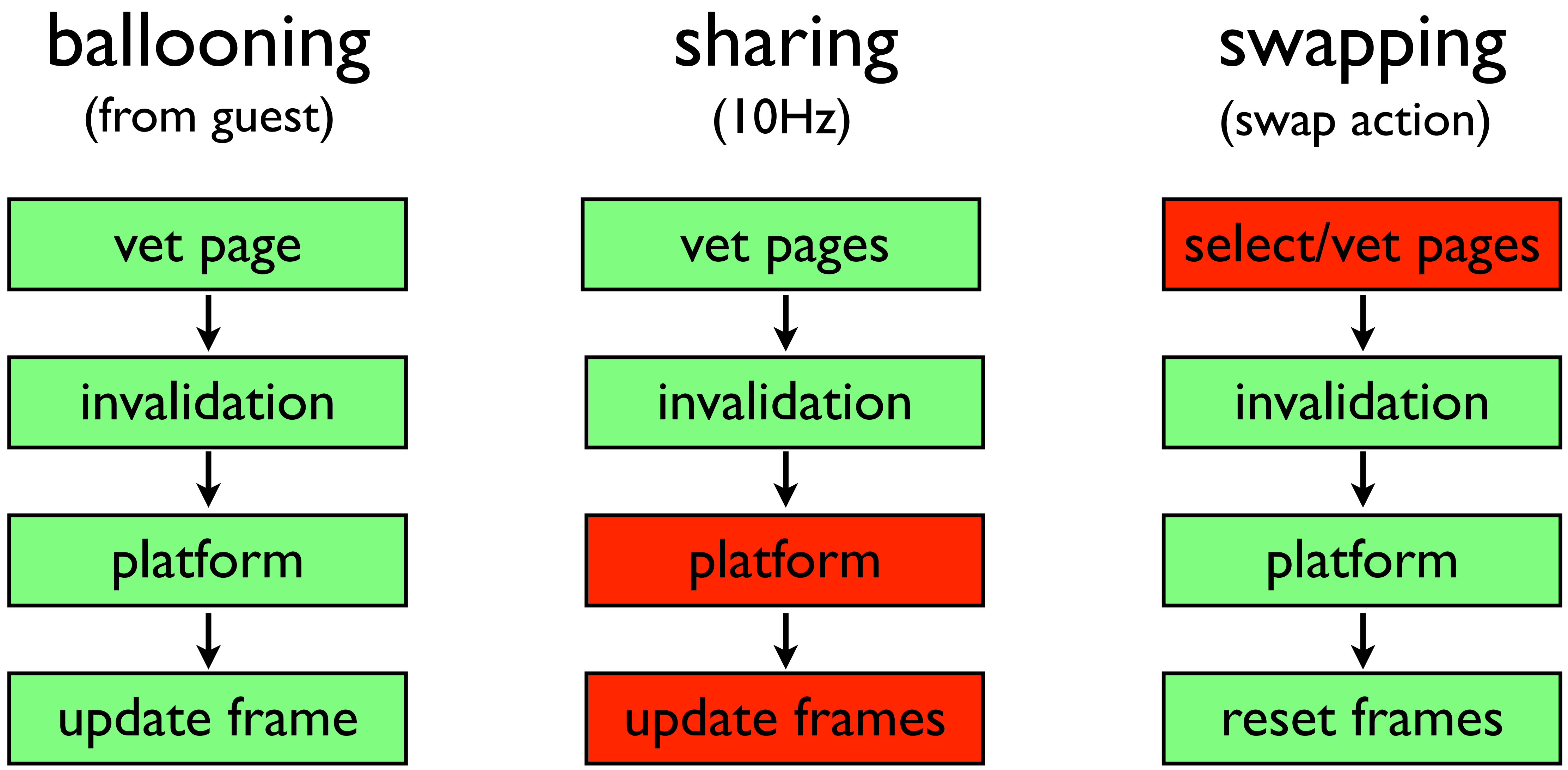
monitor-list@vmware.com

Backups

Pre-KL Services

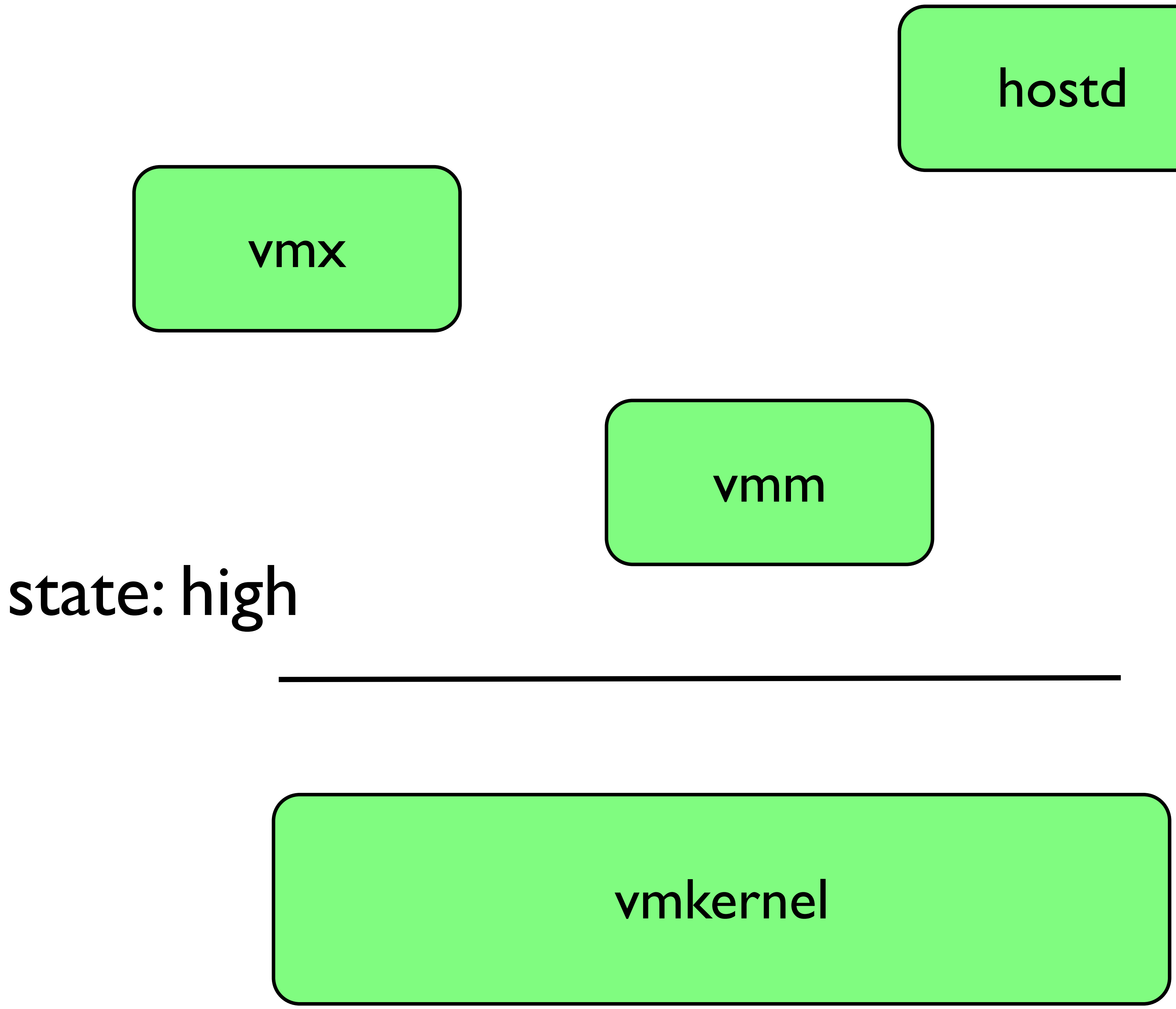


MemSched Services

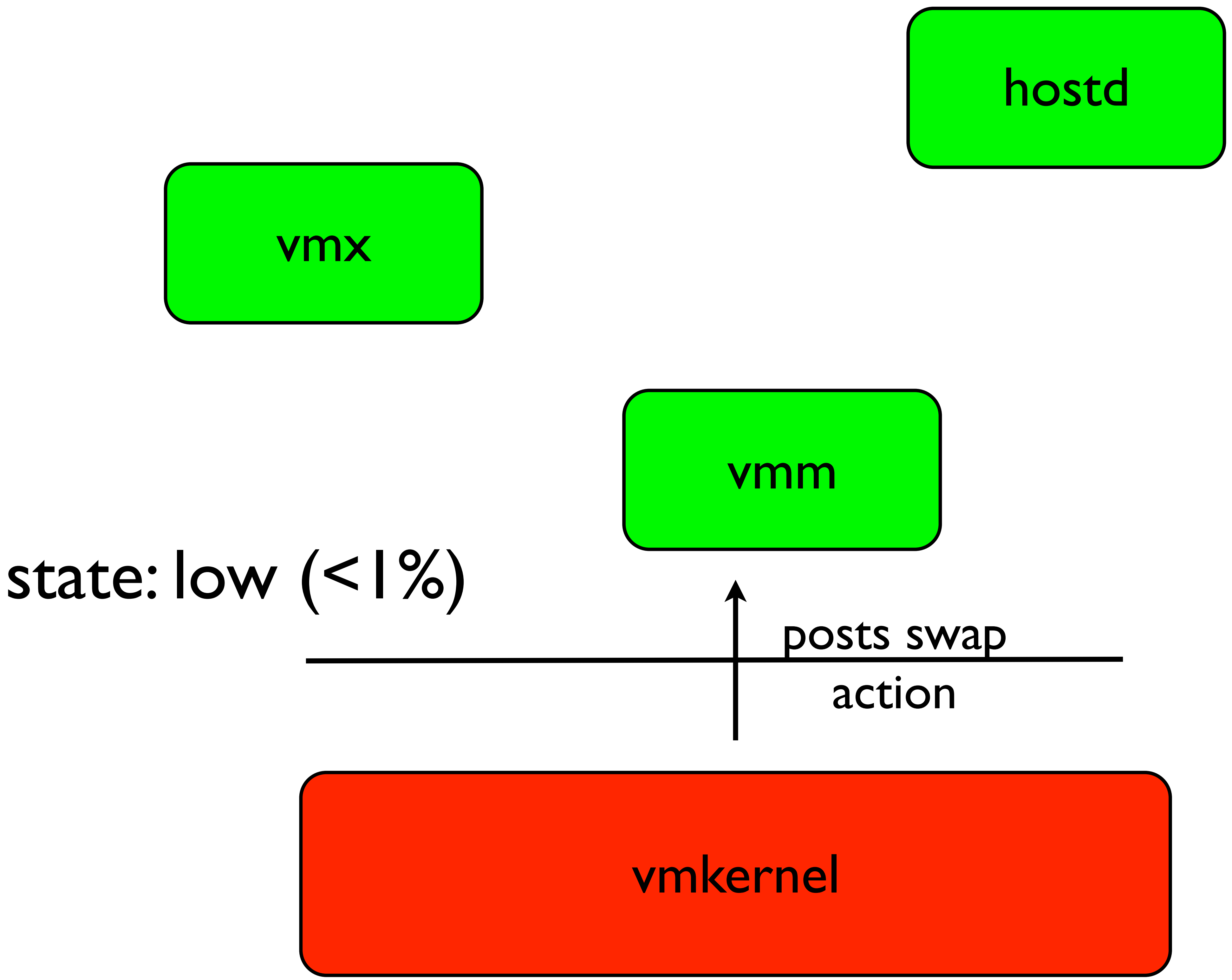


- knowledge of backing MPNs (swapping)
- cost of in-line computation (sharing)
- likelihood of access (sharing/swapping)

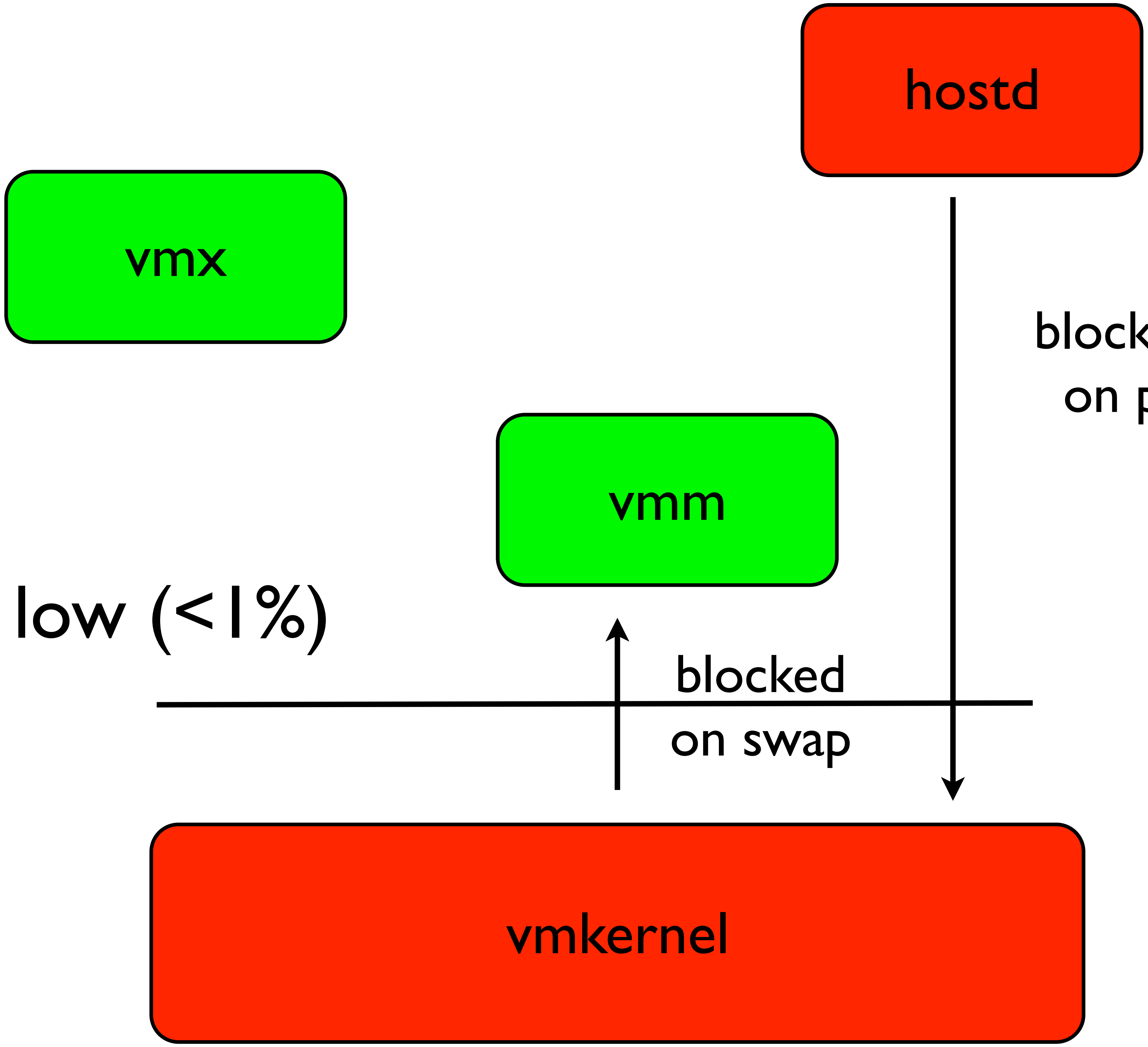
MemSched Deadlock



MemSched Deadlock

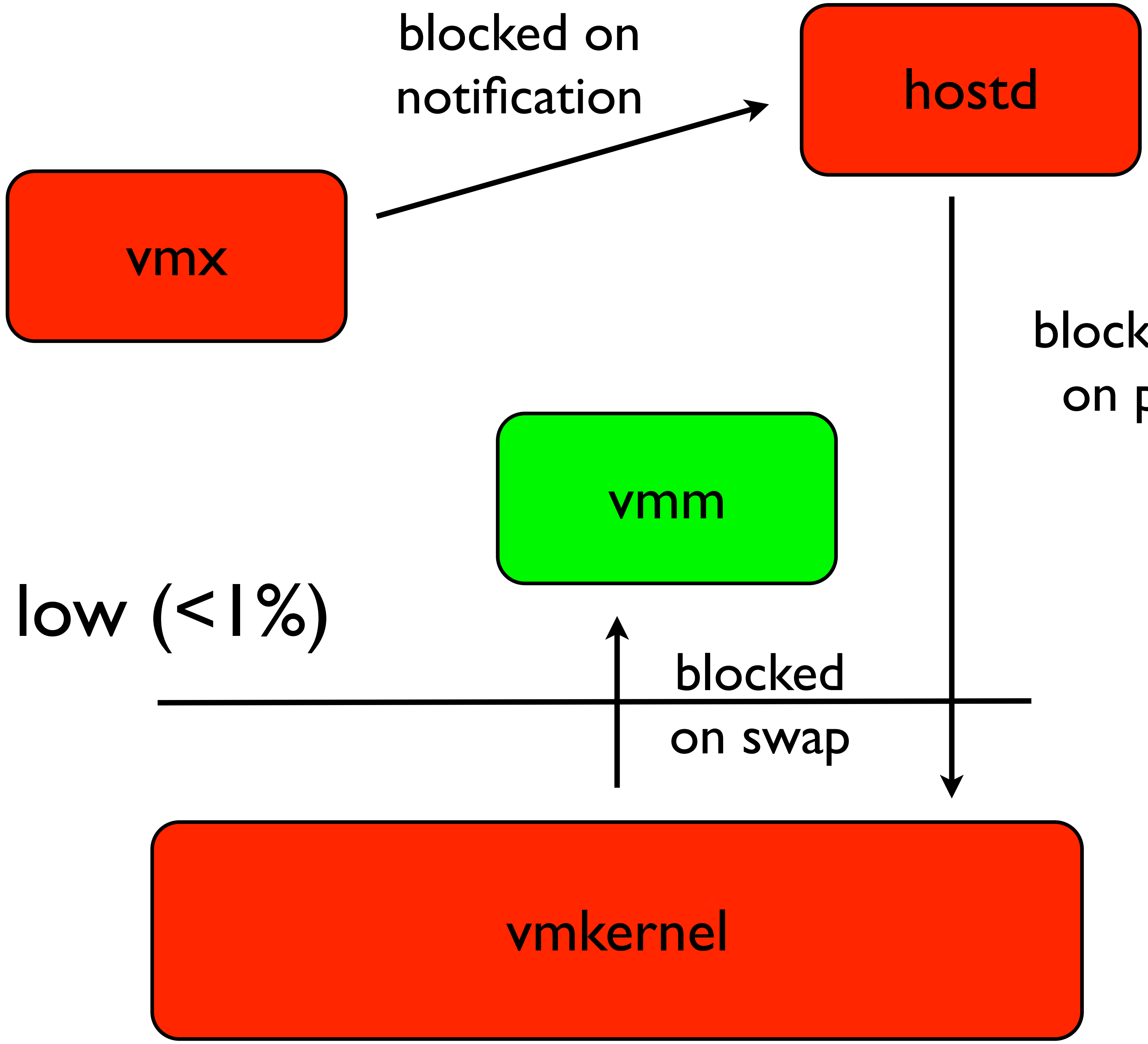


MemSched Deadlock



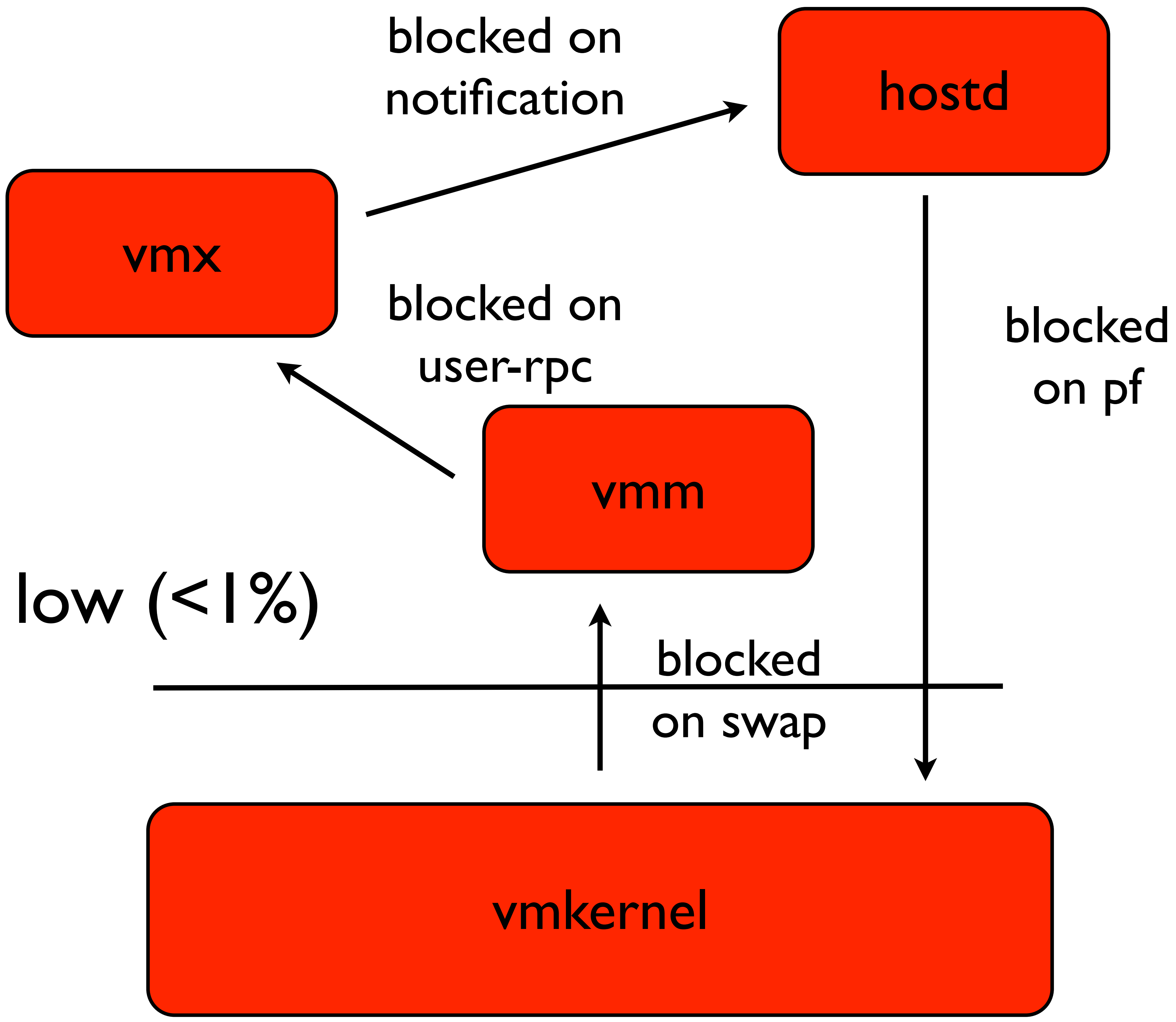
state: low (<1%)

MemSched Deadlock



state: low (<1%)

MemSched Deadlock

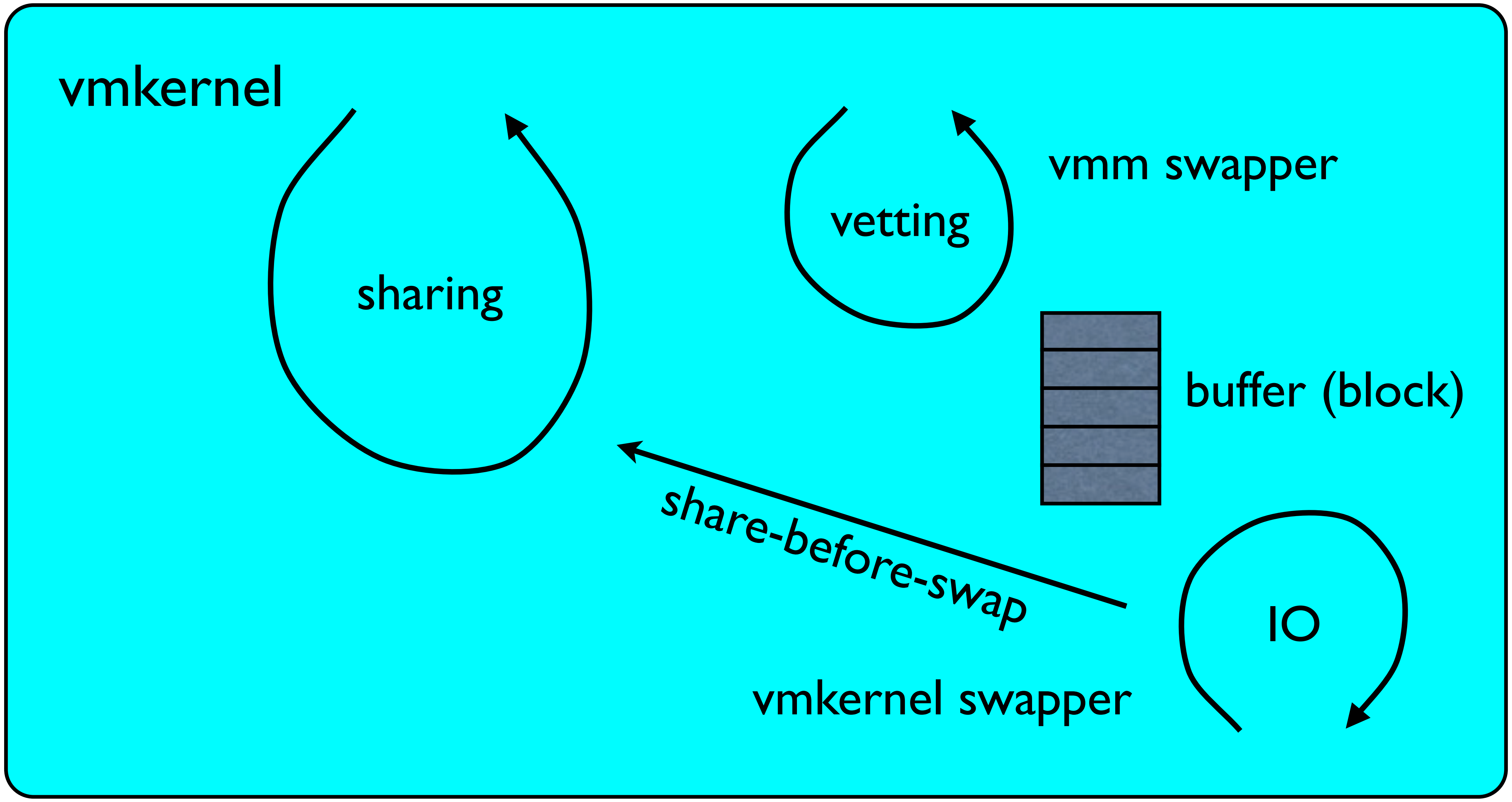


state: low (<1%)

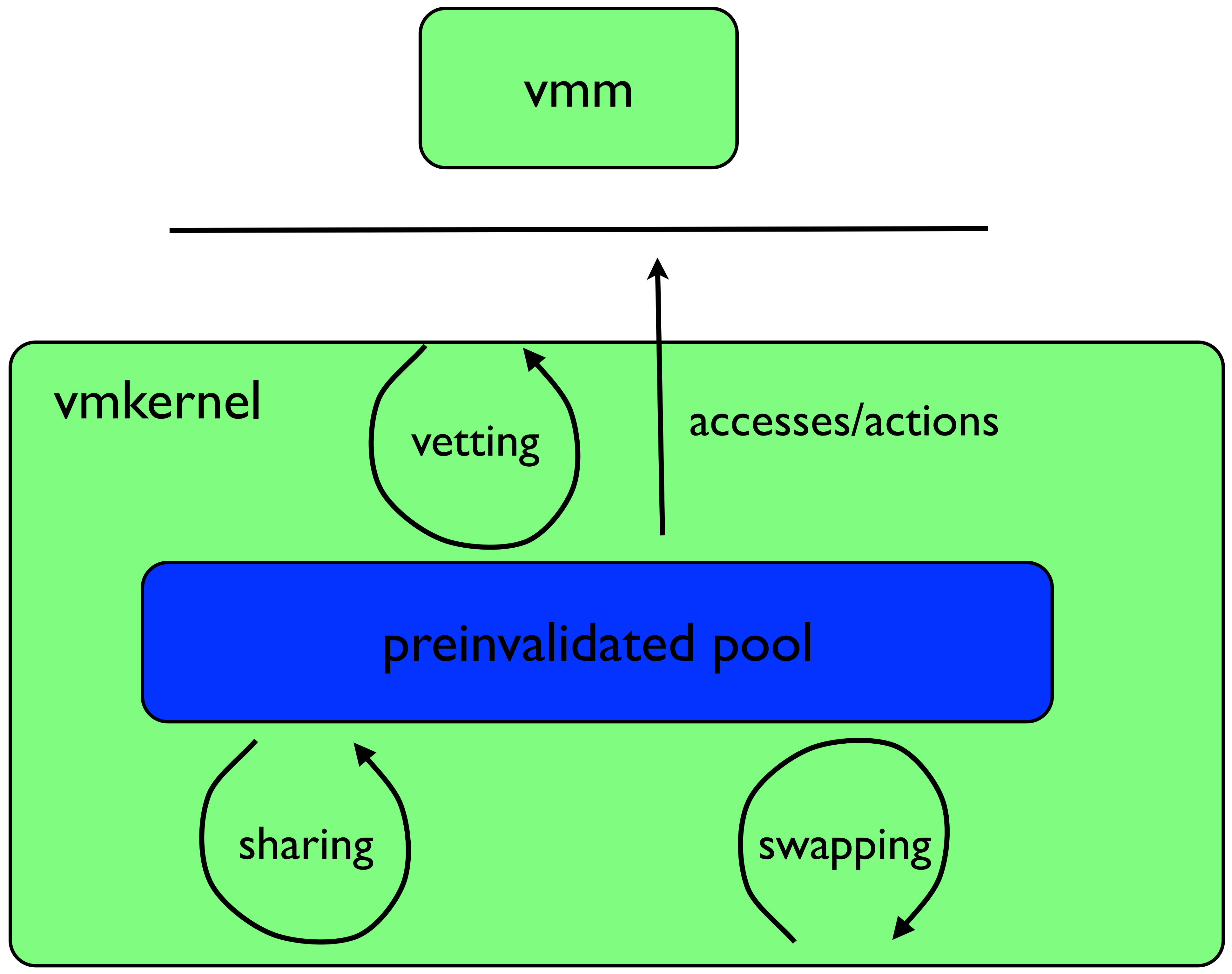
Many Solutions

- VMX-notification/vmkernel timeouts
- Allow invalidation across user-rpcs
- Vmm worker-threads
    - 1-vcpu VMs
- Privileging hostd worker threads
- Pre-invalidated/not-exposed pages

Services in vmkernel

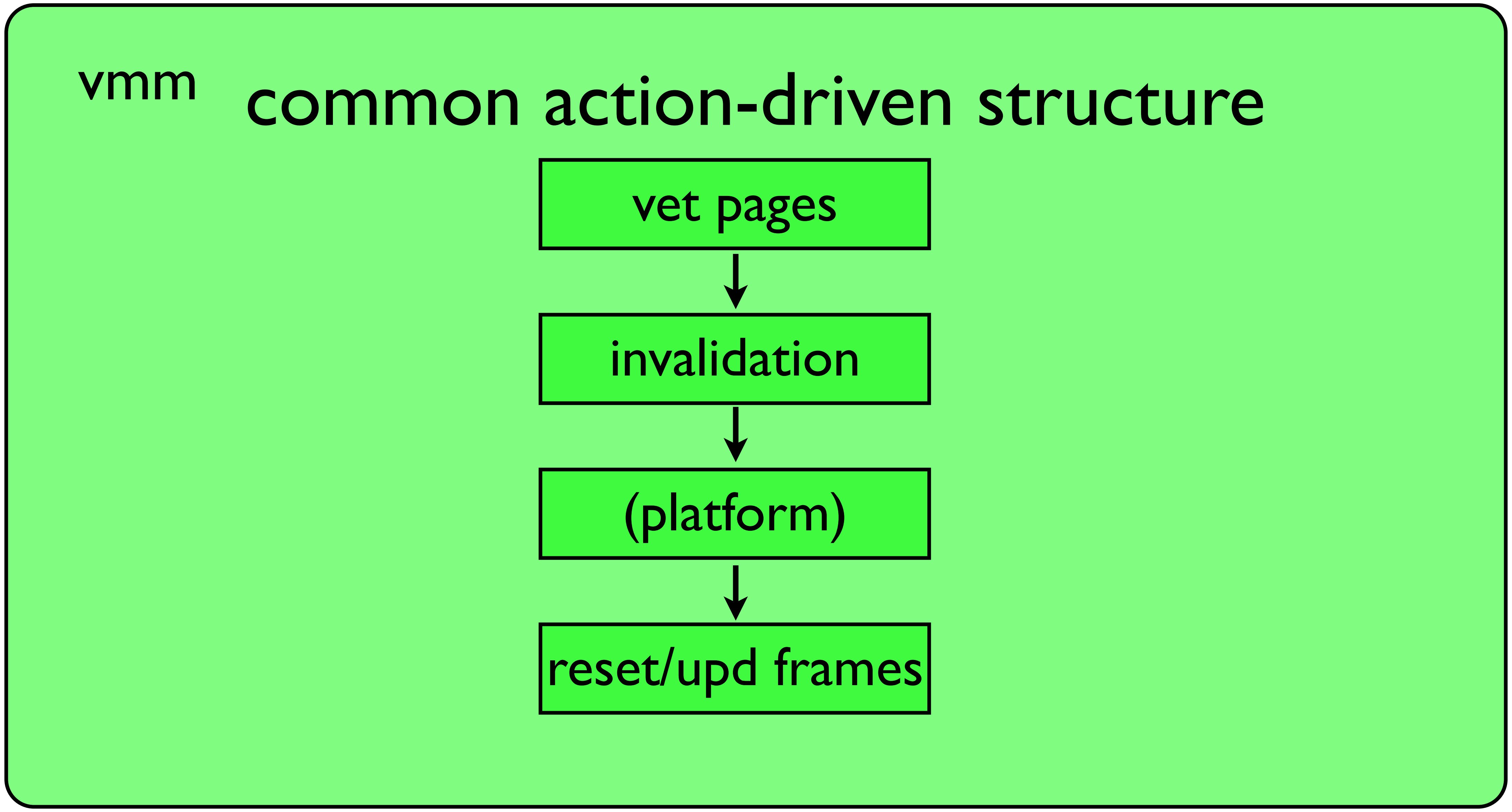


MemSched Services



!exposed
sizeable

MemSched Services



- extend to sampling, remapping (bulk validation)
- other uses: simpler vmotion, trace-installaton

What's Different

- Separation of policy/mechanism
  - table-defined, common structure
- Removal of (expensive) synchronous work
  - limit on sharing rate
- Simplification of services within platform
  - vmm/vmk-swapper, sharing v. share-before-swap
- Pre-invalidation pool
- Remapping generalized to bulk-(in)validation
- Elimination of stopping to release memory
- Guest-hints to platform