Overcommitment, VMTrack, and Preallocation

Alex Garthwaite

Monitor Group

December 17, 2012

Overview

Agenda

- today: overcommitment and how the VM (guest/vmm) respond
- Tuesday: vmmem services that release/remap memory
- Wed. AM: sampling techniques
- Wed. PM: translation, invalidation, and large-page/scaling issues

Theme: management of machine memory (MPNs)

Why the focus on overcommit/release of memory?

- MPNs exposed in vmm and vmkernel
- cooperative effort that requires time/coordination
- in past, has limited platform's ability to manipulate memory

Overcommitment

- **Scheduler**
  - tension between reservations and best-effort (shares)
  - workload consolidation
  - maintenance mode/DRS consolidation
- **Correctness/safety property: VMs should not panic**
  - ... but timing effects, out-of-memory (OOM) events
- **Strategies**
  - appBallooning (guest-driven)
  - ballooning (guest/vmm)
  - swapping to compression-cache, SSD, or disk (vmm)
- **MinFreePct reserved**
  - 6% (to as low as 1% for large VMs)
  - memory states: high, soft, hard, low
  - memsched levels across platform (not per resource pool)
- **MemSched cycle**
  - driven by MEM_BALANCE_PERIOD (15 seconds) or by change in level
  - per-VM target and consumed: target increased as delta above consumed

Overcommitment

high

soft

64% of minFreePct

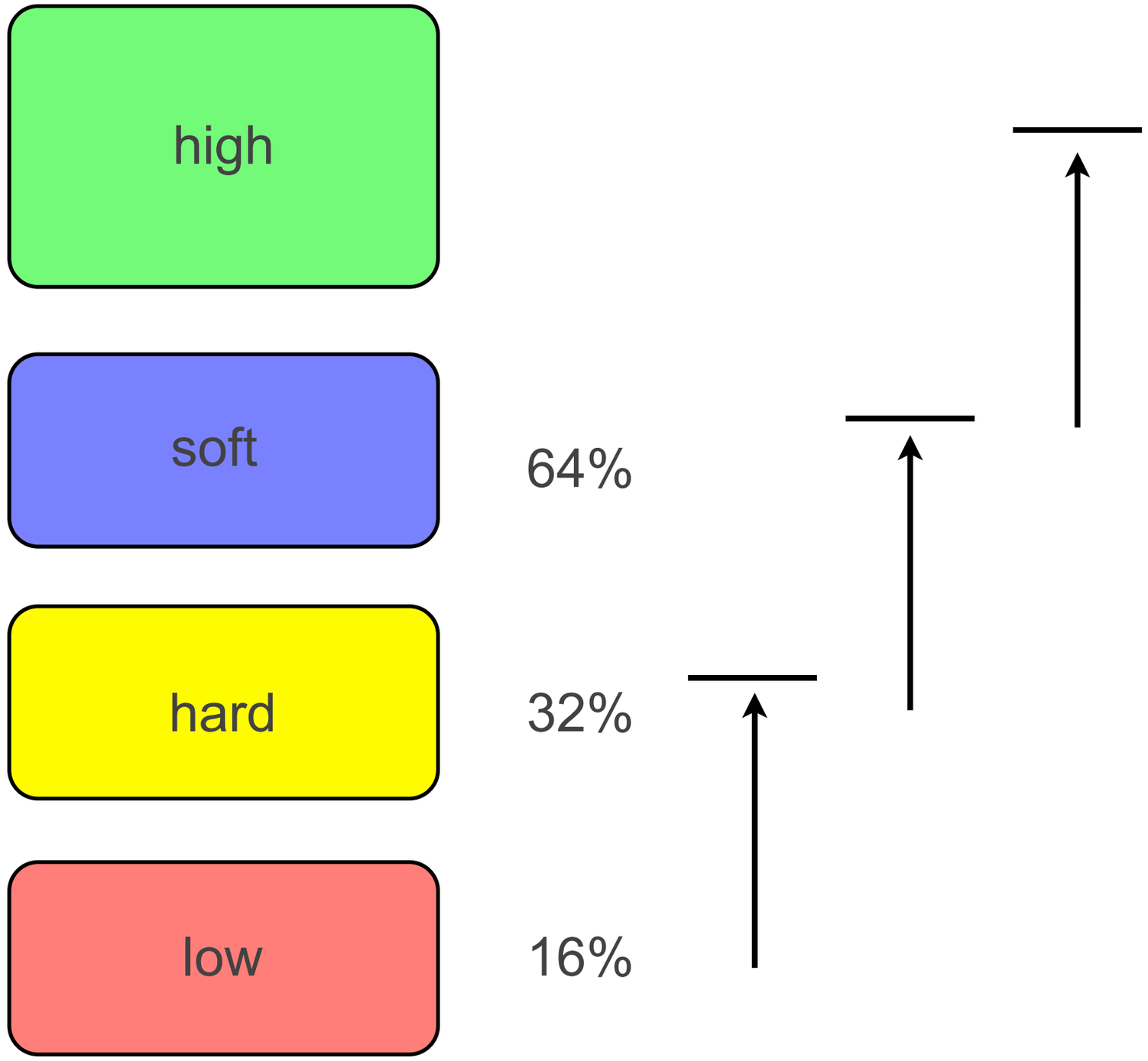
hard

32%

low

16%

Overcommitment



Overcommitment

high

soft

balloon

fall back to swap

hard

swap to zip, SSD, disk

low

repeatedly swap

Overcommitment

high	(can balloon: see pr964269)
soft	balloon fall back to swap
hard	swap to zip, SSD, disk
low	repeatedly swap

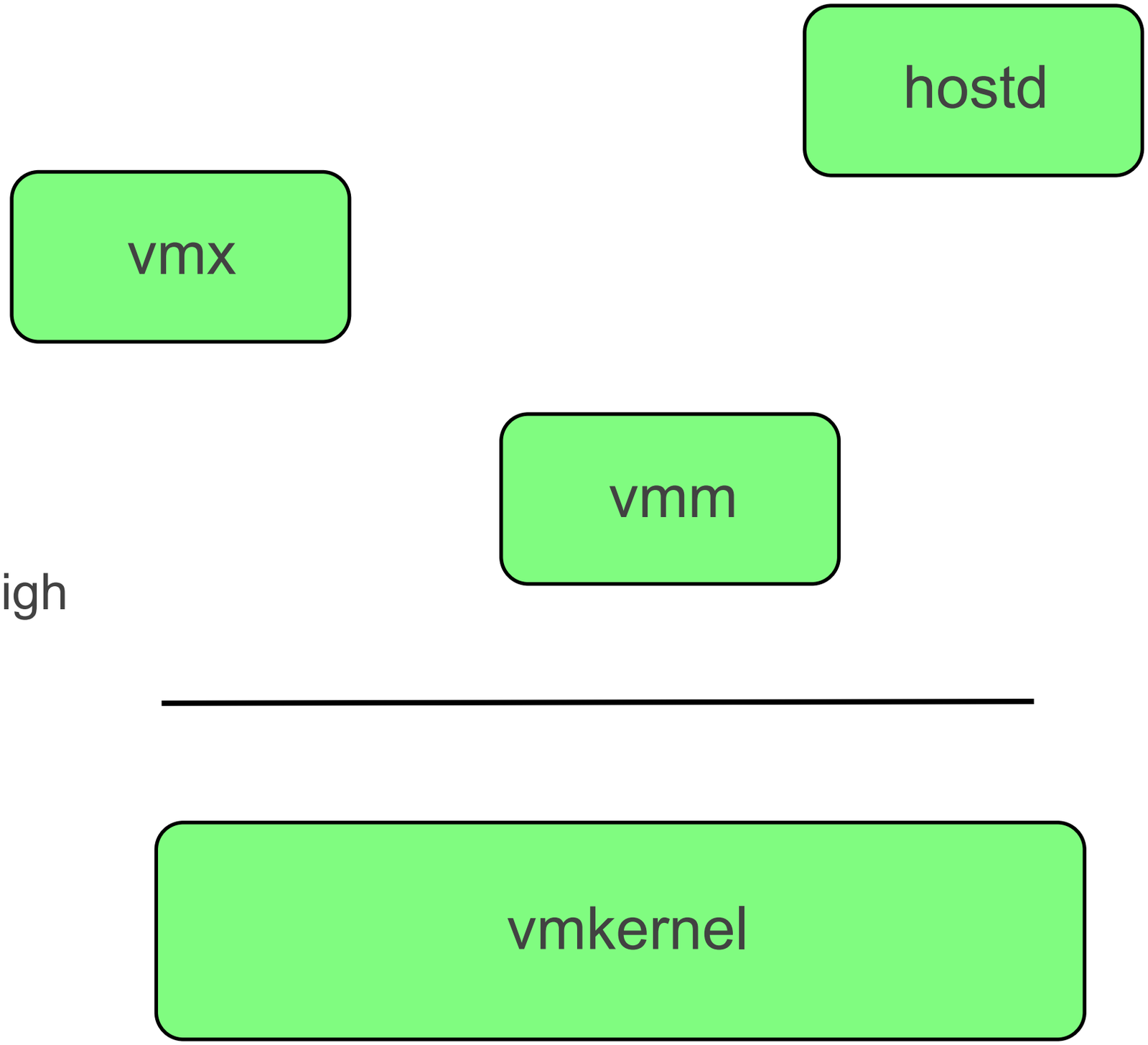
Getting Better And Better...

- **Almost all vmkernel memory now managed**
  - VMs and userworlds release
  - Question: do vmkernel heaps (as caches) have notion of back-pressure?
- **Userworld swapping (when enabled)**
  - uses LRU-replacement for selection
  - no longer need to count (most) of pageable overheads in reservation

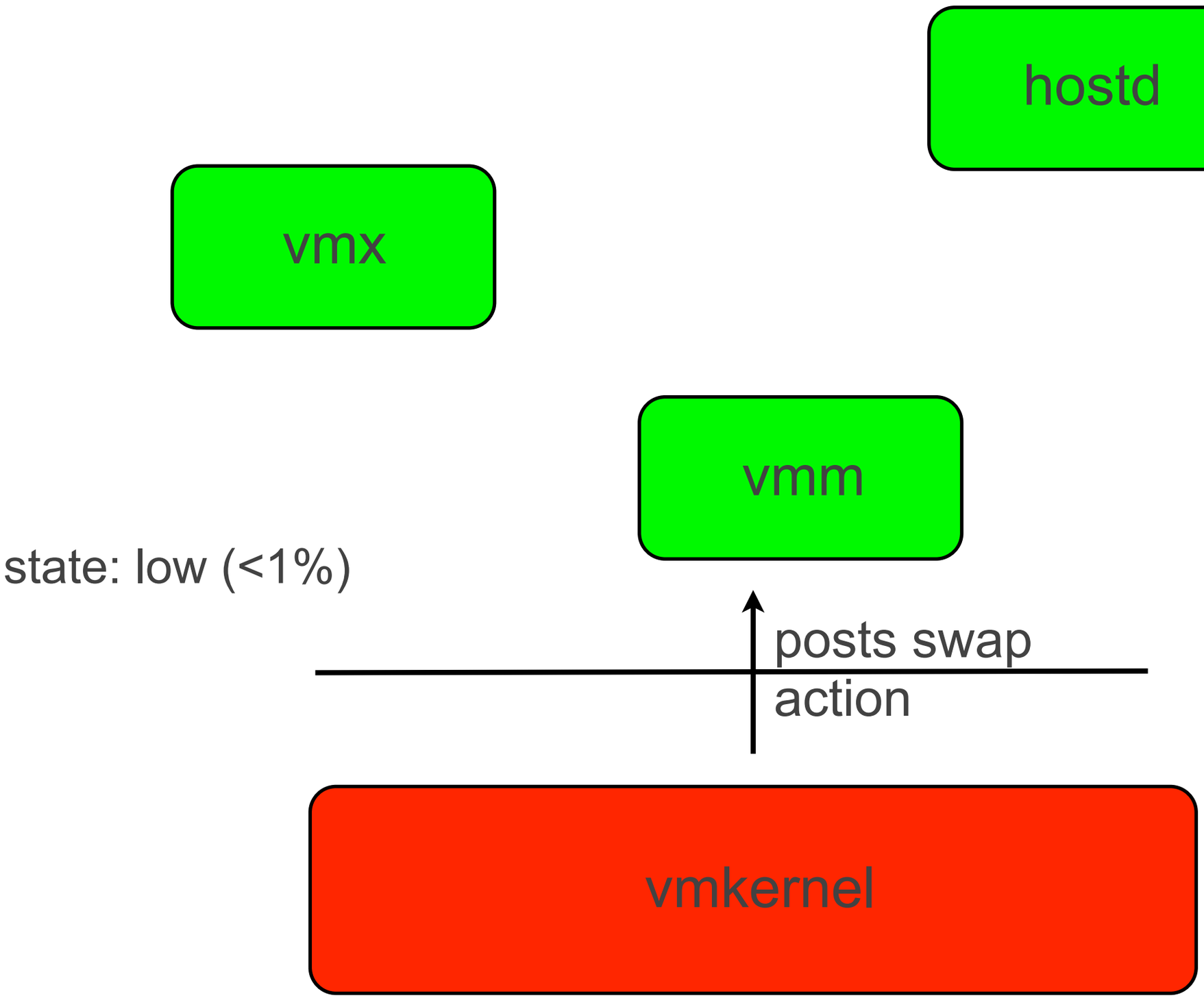
Priority Inversion

- **VMs must make progress to release memory**
  - ballooning: guestOS allocation to guest vmmemctl driver
  - swapping: vmm must make progress
- **Inability to block VMs with reclamation targets**
  - no ability to prioritize allocations (to ones with full reservation)
- **VCPUs must be in vmm for crosscalls**
- **VMs' VCPUs with no target can be blocked for duration**
  - both in hard- and low-memory states
  - faulting userworlds as well!

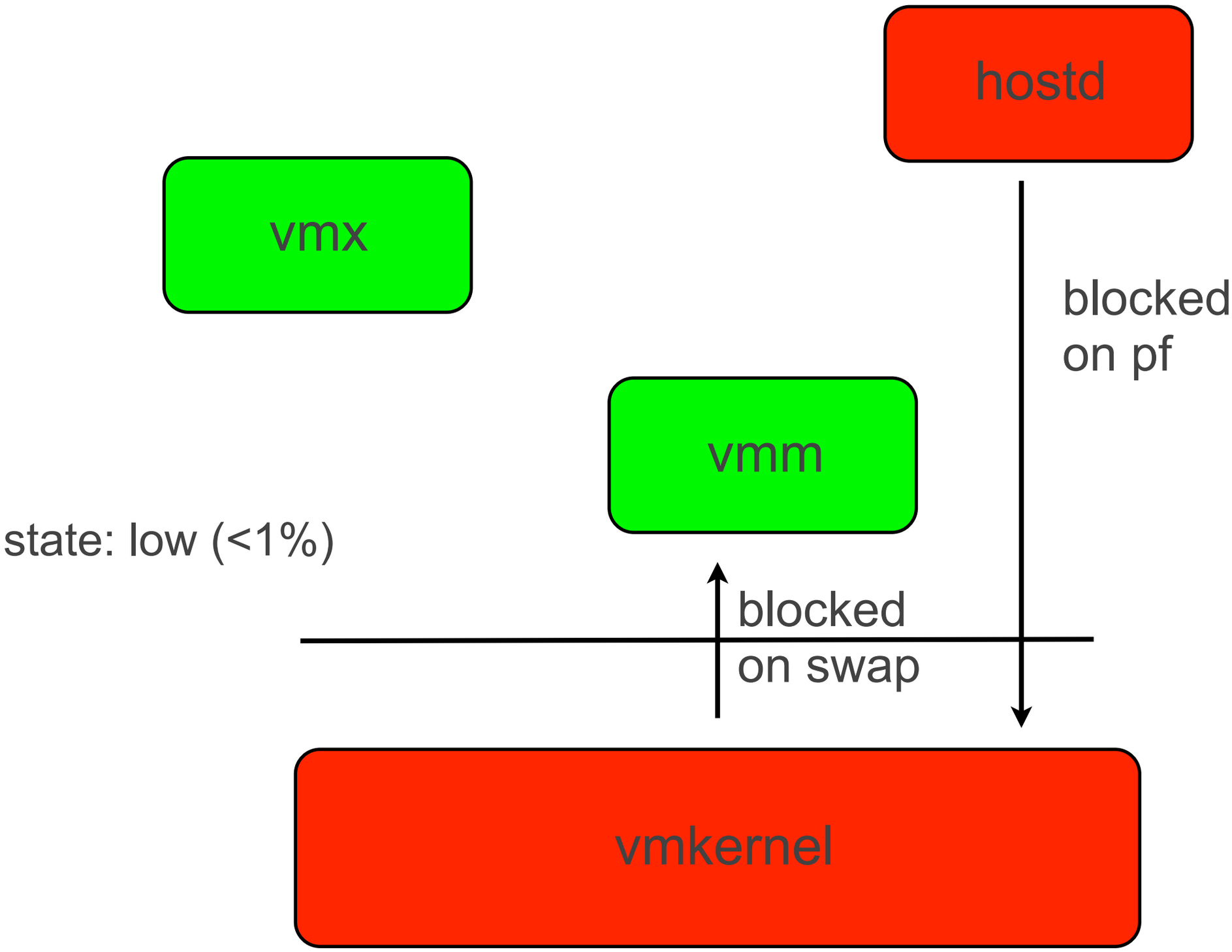
Overcommitted Deadlock



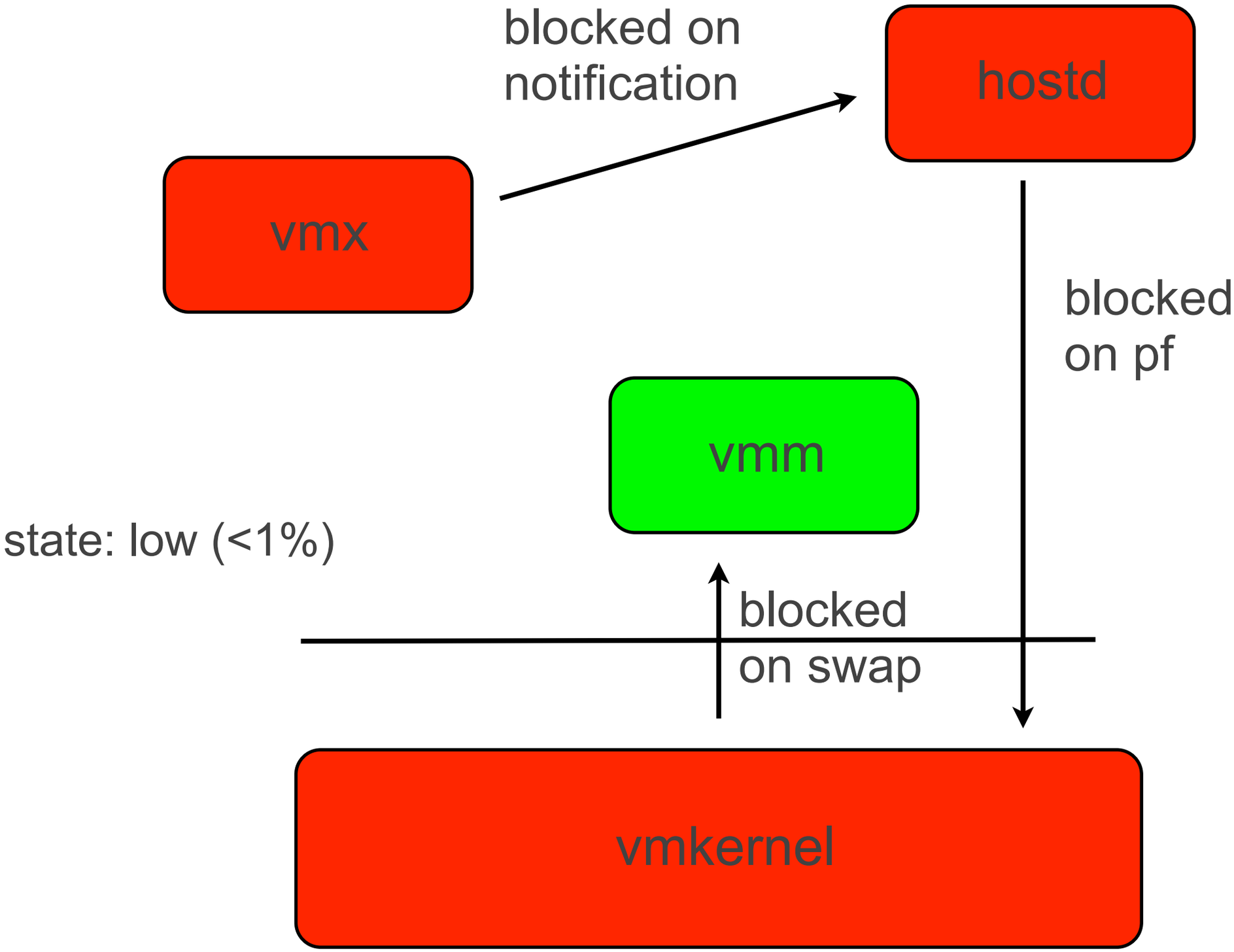
Overcommitted Deadlock



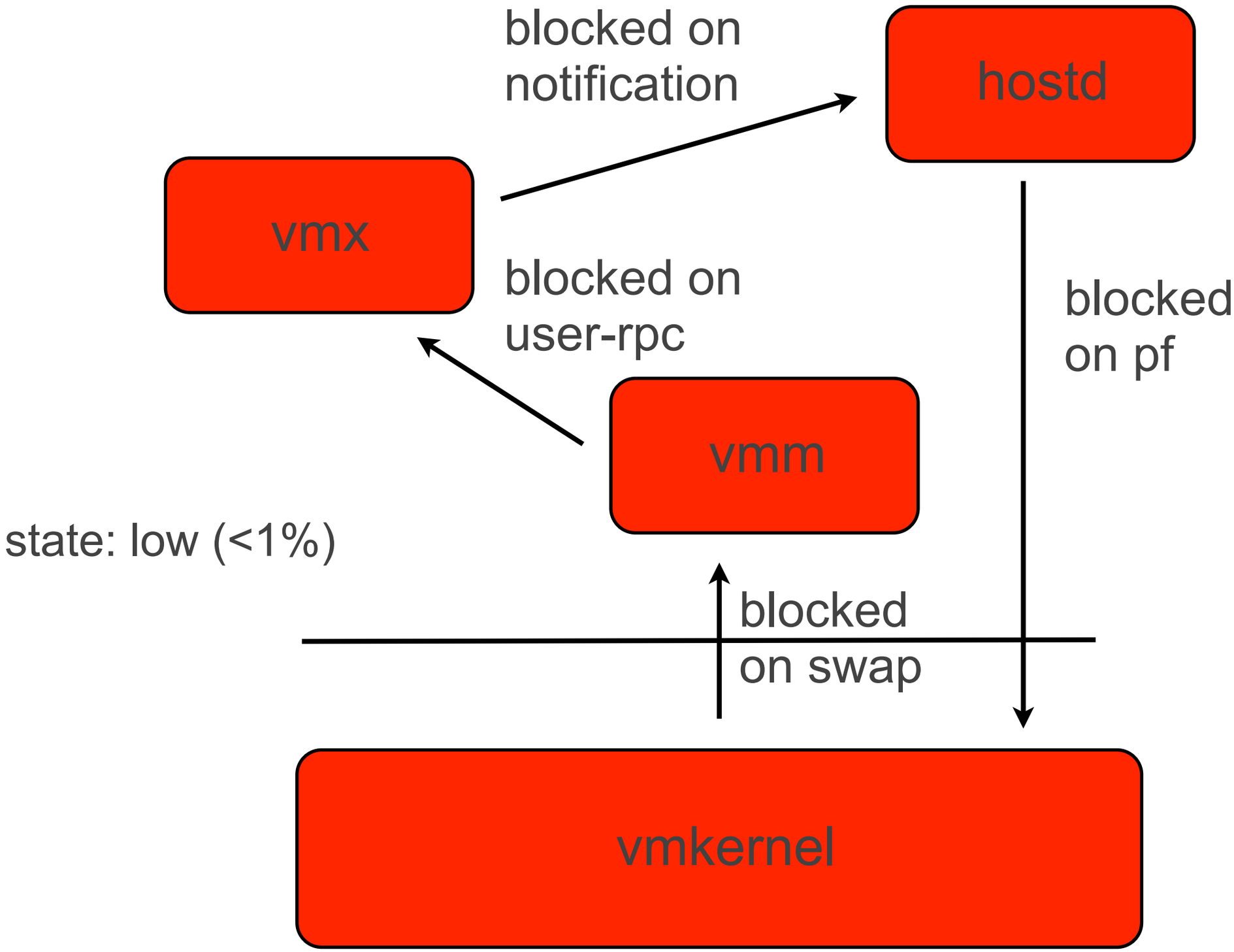
Overcommitted Deadlock



Overcommitted Deadlock



Overcommitted Deadlock



Solutions to Deadlock

- **Time outs**
  - timeout on synchronous communication between vmx and hostd
  - vmkernel timeout: unresponsive VM period (5 minutes)
- **Avoid hard/low states**
  - peg swap-rate to exceed allocation rate of VMs
    - repeated swapping in low state
    - reissue swap target on allocation in hard/low states
  - per-VM and per-VCPU allowances
    - release mechanisms (like compression) that allocate memory
    - vmx/vmkernel activity
  - VmTrack_AllocAllowance and MemSchedMaxVmmSwapTarget
    - 2x the possible allocate rate between two release points, not actual
- **Killing VMs unresponsive to swap requests**
  - VMTrack module in vmkernel: VMTrack_KillIfUnresponsiveVM
    - NB: logging from this needs to be throttled (source of PSODs)
  - vmm VMMem checks and action in monitor to handle release requests
  - fall back to time-out

Swapping and VMTrack Before ESX 4.1/5.0

- **Assumed small amount of allocation per guest instruction**
- **Only handled release-requests through actions**
  - easy to reason about where releases could happen
  - bounded swap-costs per instruction
- **Old limits simply per VM: problem of many vcpus**
- **Provided little insight into VMTrack failures**
  - per-VM check on anon and guest-page allocations
  - could not tell whether cause was in vmm, vmx, or vmkernel
- **Requirement to make reasoning about allocation behavior as local as possible**

VMMem/VMTrack Mechanisms

- **Enabled whenever vmm swapper is enabled**
  - Platform_MonitorStarted, around checkpoint events
- **Limits per-VM and per-vcpu bounds**
  - also, e.g., compression cache, pending swap-list
- **Per-vcpu enforcement: VMMem module in monitor**
  - source files
    - vmmem_int.h: internal prototypes, e.g., for checking for release requests
    - vmmem_ext.h: functions to enable/disable release checks
    - vmmem.c: main code
  - primary functions
    - VmMem_AllowReleasesOnAllVcpus, VmMem_DenyReleasesOnAllVcpus
    - VmMem_AllowReleasesByVcpu, VmMem_DenyReleasesByVcpu
    - VmMem_CheckAcquiredRankForReleases, VmMem_CheckReleasedRankForReleases
    - VmMem_CheckForReleaseRequests
  - structures/counters
    - vmmDenyReleasesOnAllVcpus (stopped), vmmDenyReleasesByVcpu
    - vmmemCheckCount

VMMem/VMTrack Mechanisms

- **Denying Zaps (and Releases)**
  - all vcpus
    - monitor migration
    - GPhys/busmem migration
    - MMU structure resizing
  - per vcpu
    - unpinned MPNs through translation
    - locks of too high rank
    - interrupts disabled
- **Checks by page type**
  - guest pages: must allow zaps
  - anon pages: Alloc_Init/Alloc_Alloc/AllocWork and AllocGetAnonPagesFromPlatform
  - note also: VmMem_ReleasePages
- **Programming idioms**
  - allocate pages, then deny zaps
  - explicit poll points: VmMem_CheckForReleaseRequests
- **VmMemCheck count reset when actions drained**
- **Current bound is 48 pages per vcpu (primarily due to numa-mig)**

Incomplete Support for VMTrack

- **Sequences of allocations when releases denied**
  - currently only checked at poll-points, allocation sites
  - better to check when reasons for denying releases cease
    - performance cost? vs. much simpler reasoning about bounds
- **VMX and vmkernel allocations**
  - device-related allocations
  - either handle failure or allow blocking
  - SVGA thread in vmx does latter
  - but currently no bound, can cause VMTrack to kill VMs

Preallocation

- **PRs 720036, 733949 and 908410**
  - VSA and Nutanix storage
  - deadlock if vmm swap files managed by vApps
- **Memory preallocation**
  - sched.mem.prealloc.mainMem: full reservation, no sharing
  - sched.mem.prealloc.pinnedMainMem: also, no remap/retire
  - sched.mem.prealloc: prealloc overhead/anon and mainMem
  - must separately disable vmx swapping: checked at start
- **True preallocation**
  - done in Monitor_MonitorStarted just before running guest
    - SMP-FT has hook to complete its set-up here as well
  - reservation increases feed prealloc pool or fail
  - will not block even under low-memory conditions
- **Anon pool size based on reservations**
  - initial VM admission estimate
  - reduced in OvhdMem_LatePowerOn just before starting the vmm

Flow-Based Scheduling: Less like a sump pump

- **Current strategies**
  - speed up reclamation (e.g., batched ballooning)
  - plan to slow allocating contexts to match reclamation rates
- **Problem: bounding allocation in vmx by vcpus**
  - SVGA module as example: allow blocking as needed
- **Basic observation: reclamation takes time**
- **Measure rates of reclamation to gauge exhaustion**
  - support small minFreePct levels
  - anticipate reservation-based allocation rates from VMs

Recap

- **Short term**
  - track impact of release-request points (stats/vprobes)
    - time spent on release per instruction
  - change VMMem polls to check when swap possible again
    - code bloat? performance impact?
  - fix monitor NUMA migration to not disable zapping (reduce per-vcpu VMTrack bound)
  - fix max swap target to be a function of actual allocation rate
  - improve rates of reclamation (ballooning: batched, 2MB, sharing)
  - slow allocation rates of VMs w/o reclamation targets
  - find better balance between sharing and large pages (Wed. AM discussion)
  - refactor/clean up hosted and vmkernel config set-up of MemSchedInfo in MemSched_PowerOn
  - throttle logging from VMTrack module to prevent PSODs

Recap

- **Long term**
  - reduce vmm/vmx dependency (more on Tuesday: vmmem services)
    - not-exposed pool: requires
    - preclean dirty pages from not-exposed pool
    - remember already swapped contents
  - bound per-vcpu vmx allocation limits

Recap

- **Longer term (more on Wednesday: invalidation)**
  - allow vmkernel/platform to manage pagetables
    - most MPNs not cached outside of GPhys or sw-mmu
    - reclaim most pages without vmm handshake
  - additional levels/priorities for allocation (hostd, vApps)
  - flow-based scheduling to avoid performance cliffs
    - base minFreePct on reclamation rates
    - better anticipate needs of reservations
  - revisit applying memsched states to MemSched resource-pool hierarchy