Virtio-(balloon|pmem|mem): Managing Guest Memory

David Hildenbrand & Michael S. Tsirkin
Red Hat

Motivation: Manage Guest Memory

Speed up migration

- Any VM memory possibly contains “important” data
- Some VM memory contains data that is not worth migrating

Reduce host swapping when overcommitting memory

- There might be a lot of unused / free memory inside VMs
- Instead of swapping, rather temporarily “steal” unused memory from VMs

Control / Shrink the pagecache in the VM

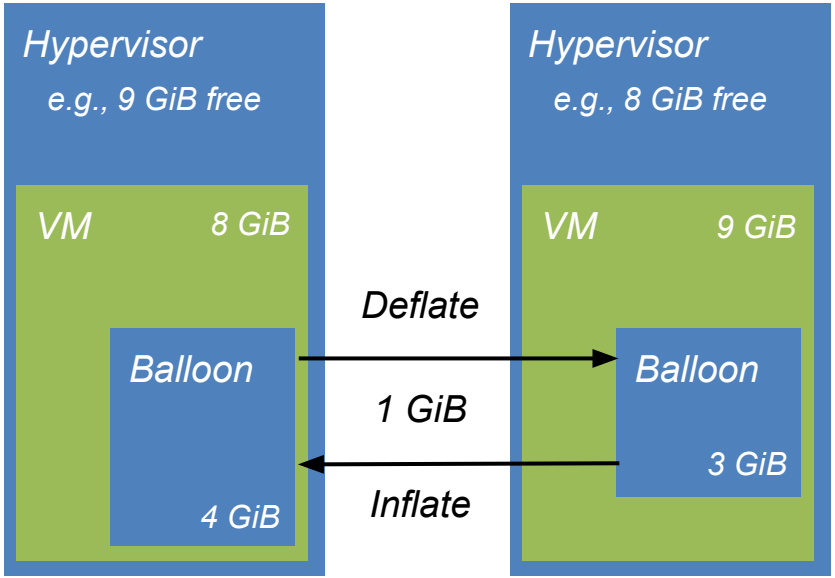
- Guest OS in the VM will try to make use available memory for caches.
- Some data in caches can be dropped without affecting workloads

Dynamically resize VM memory - memory hot(un)plug

- Automatically (VM needs) or manually (user requests)

Recap: Memory Ballooning

“Relocate physical memory between a VM and its hypervisor”



Balloon Inflation

- Guest driver allocates memory and tells the hypervisor about it
- Hypervisor can reuse inflated memory (e.g., for other VMs)

Balloon Deflation

- Guest driver frees previously allocated memory after telling the hypervisor
- Guest OS can use deflated memory

Controlled by “target balloon size”

- Requests to change “logical VM size”

Red Hat

THE LINUX FOUNDATION

Using Memory Balloon Inflation / Deflation ?

Speed up migration

- Any VM memory possibly contains “important” data
- Some VM memory contains data that is not worth migrating

inflate balloon before migration + deflate balloon after migration

Reduce host swapping when overcommitting memory

- There might be a lot of unused / free memory inside VMs
- Instead of swapping, rather temporarily “steal” unused memory from VMs

dynamically inflate/deflate balloon

Control / Shrink the pagecache in the VM

- Guest OS in the VM will try to make use available memory for caches
- Some data in caches can be dropped without affecting workloads

inflate balloon + deflate balloon

Dynamically resize VM memory - memory hot(un)plug

- Automatically (VM needs) or manually (user requests)

dynamically inflate/deflate balloon

Without going into detail, there are a lot of issues ...

Virtio-(balloon|pmem|mem)

Speed up migration

- Any VM memory possibly contains “important” data
- Some VM memory contains data that is not worth migrating

Reduce host swapping when overcommitting memory

- There might be a lot of unused / free memory inside VMs
- Instead of swapping, rather temporarily “steal” unused memory from VMs

Control / Shrink the pagecache in the VM

- Guest OS in the VM will try to make use available memory for caches
- Some data in caches can be dropped without affecting workloads

Dynamically resize VM memory - memory hot(un)plug

- Automatically (VM needs) or manually (user requests)

~~*inflate balloon before migration + deflate balloon after migration*~~
virtio-balloon: free page hinting

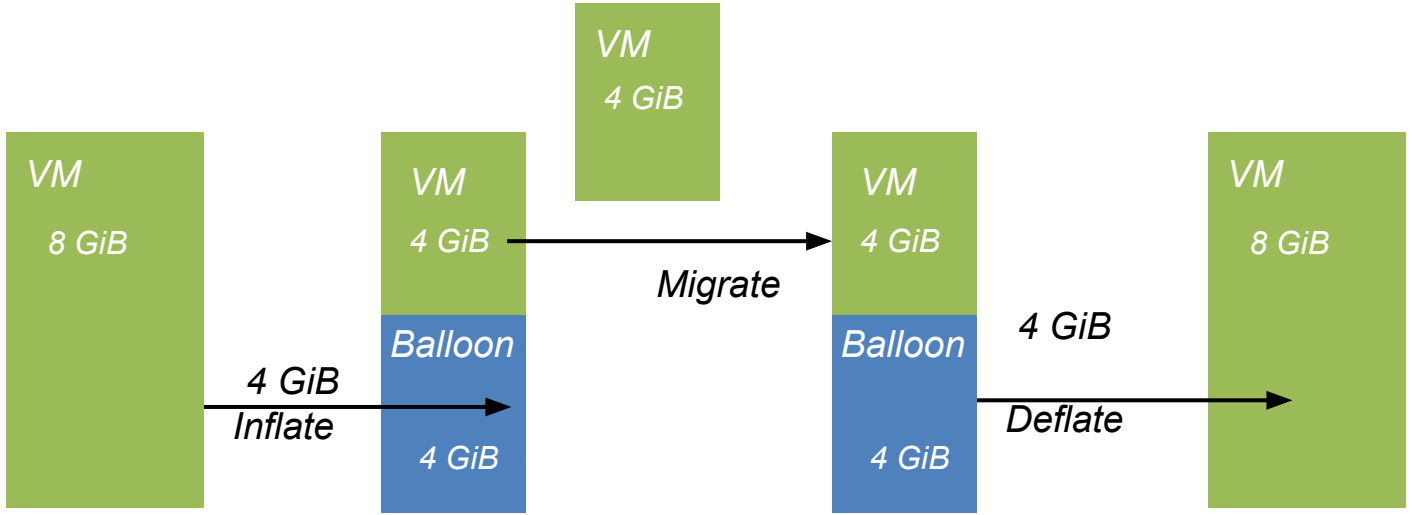
~~*dynamically inflate/deflate balloon*~~
virtio-balloon: free page reporting

~~*inflate balloon + deflate balloon*~~
virtio-pmem

~~*dynamically inflate/deflate balloon*~~
virtio-mem

virtio-balloon: inflate/deflate under host control

Example: migration



Inflate by how much?

- Too much slows down guest
- Not enough slows down host/migration

We can check guest stats but they change!

Red Hat

THE LINUX FOUNDATION

virtio-balloon: inflate/deflate under guest control

Idea 1: inflate up to all free memory

Don’t slow down the host

Idea 2: let guest deflate any time

Don’t slow down the guest

Idea 3: free memory is written to by guest

- Before use
- Host can detect it

No explicit deflate

Idea 4: do not fragment guest memory

- MAX_ORDER - 1 pages

Less overhead

AKA free page hinting/reporting

free page hinting: by Wei Wang (Intel)

Host: RAM write-protected (guest writes tracked)

Host: request free page hints from guest → *Guest: add all free pages to balloon*

Host: start migrating RAM

Guest: send each page to host →

Host: receive a free page hint

Guest: can shrink balloon at any time

Host: was page written to meanwhile?

- If not mark it as already migrated
- Won't be migrated unless written to later

Guest: use a free page by writing into it →

Host: detect a page fault, mark page for migration

free page hinting: pros and cons

Pros

No overhead until requested

- E.g. only during migration
- Cancel/restart any time

Reuse hypervisor write tracking

- Exists for pre-copy migration

Shrinks balloon without waiting for host

- No allocation stalls

A good fit for migration

Cons

Needs to be requested by host

- *Not always clear when*

One shot

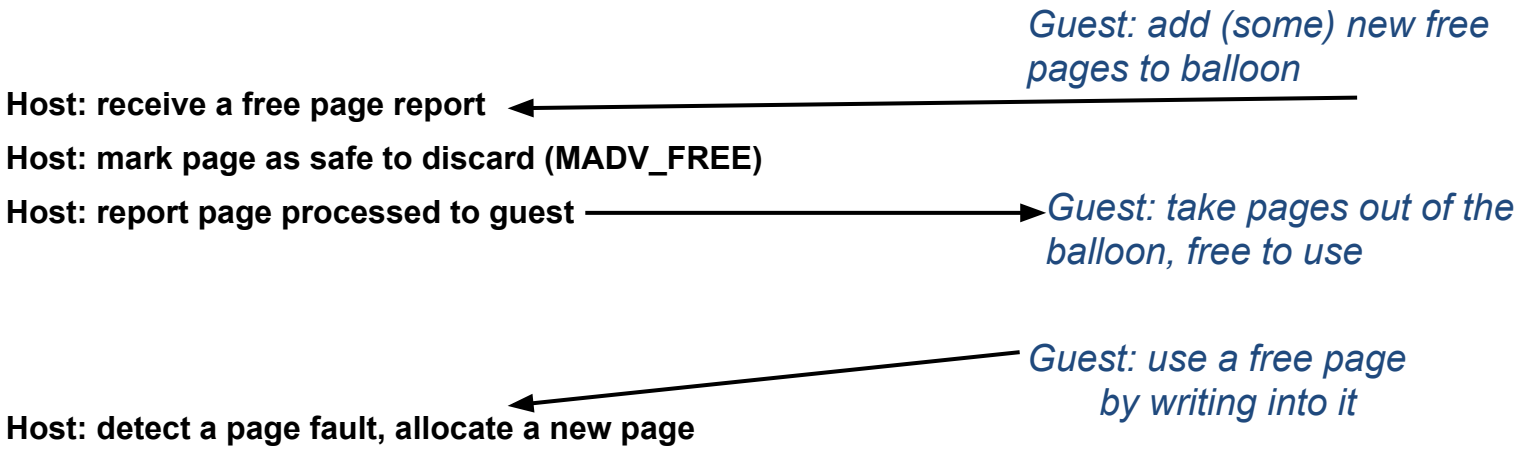
- *Inflating often - expensive*

Relies on write tracking

- *Adds overhead to writes*

Less of a good fit for overcommit

free page reporting: by Alexander Duyck (Intel)



free page reporting: pros and cons

Pros

Active at all times

- No need to activate after boot

Robust, simple host implementation

No need to track guest writes

- Potentially, less exits when memory is static

A good fit for overcommit

Cons

overhead incurred periodically

- *1-2% in memory intensive workloads*

shrinking waits for host

- *might stall because of host scheduler*

Less of a good fit for migration

Balloon: TODOs

Shrinking guest caches

- virtio-pmem is one solution for the page cache
- Application caches?

No support for VFIO

- Needs host IOMMU support

Fix inflate/deflate interface bugs

- More than 2^44 for inflate/deflate
- Different host/guest page sizes
- Spec deflate on OOM

Contributions welcome!

virtio-pmem: Overview (1)

“Emulated NVDIMM with a paravirtualized flushing interface”

Instead of virtio-blk …

“… -drive file=*./disk.img*,format=raw,if=virtio …”

… use virtio-pmem:

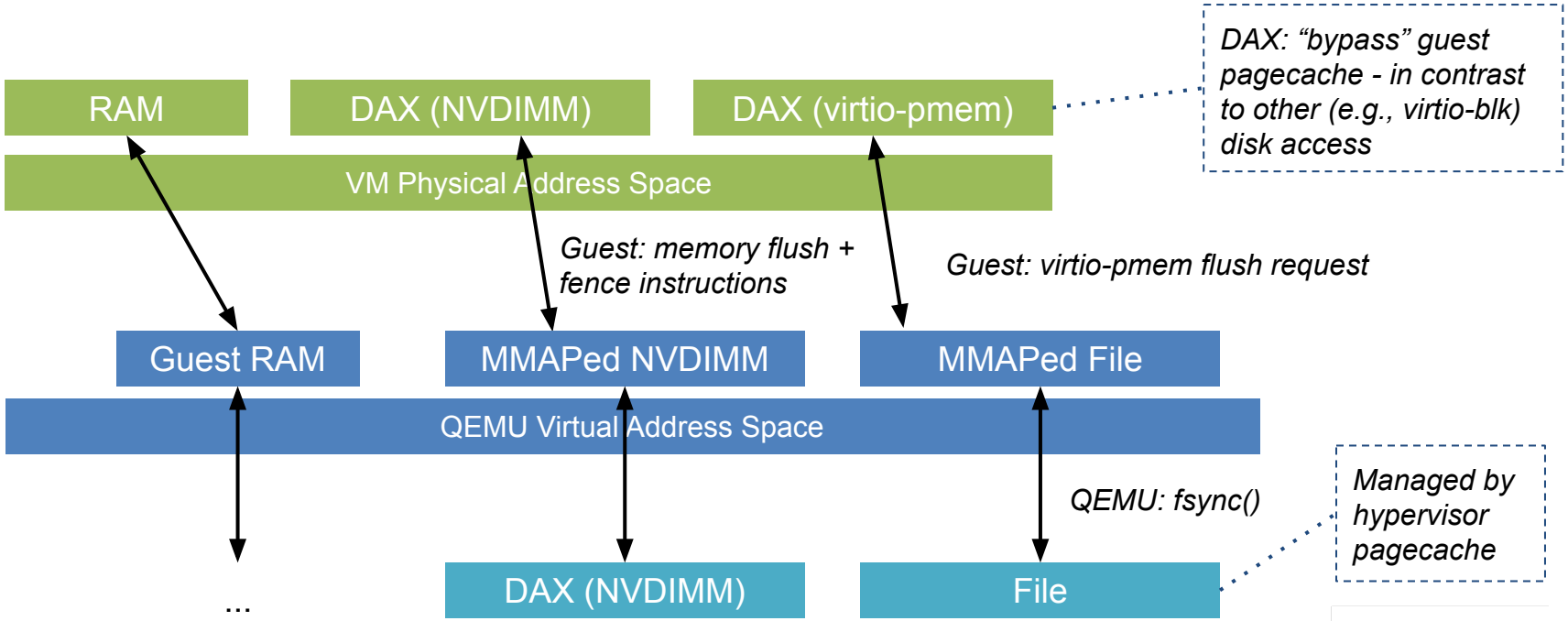
“… -object memory-backend-file,id=mem0,share=on,mem-path=./disk.img,size=4G, …”

“… -device virtio-pmem-pci,id=vpmem0,memdev=mem0 …”

Map a file (disk image) into VM physical address space

- Guest accesses disk similar to a NVDIMM (DAX), however flushes work properly
- Idea from Rik van Riel, implemented by Pankaj Gupta

virtio-pmem: Overview (2)



Red Hat

THE LINUX FOUNDATION

virtio-pmem: (Dis)Advantages and Open Items

Advantages

- Move pagecache handling from guest to hypervisor - free up guest pagecache
- “Safe” file-backed emulated NVDIMMs - writes properly flushed
- NVDIMM-like mechanism even for architectures without hardware NVDIMMs (and no ACPI)

Disadvantages

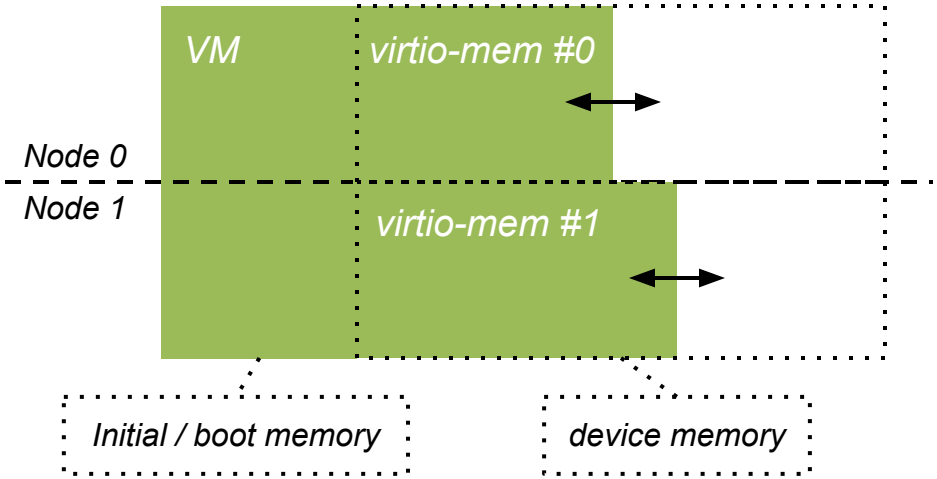
- Supports only RAW disk images for now
- Security/fairness concerns (e.g., pagecache side-channel attacks)
- Booting requires external kernel in QEMU (or other disk)
- Not applicable in all setups (e.g., if the hypervisor pagecache isn’t involved - SR-IOV/mdev, big disks?)

Open Items

- Support other architectures (e.g., arm64, ppc64, s390x) and guest OSs (e.g., Windows)
- Support other disk image types (e.g., using userfaultfd)
- Proper asynchronous flushing in Linux (to fix a preflush order issues - WIP)
- Libvirt integration, live migration, (hot)unplug support, QEMU optimization (e.g., io_ring) ...

virtio-mem: Overview (1)

“Fine-grained, NUMA-aware memory hot(un)plug to dynamically resize VMs”



virtio-mem devices

- Provide a flexible amount of memory to a VM
- Each device manages a dedicated memory region in VM physical address space
- Each can be assigned to a NUMA node
- Memory ***not*** touched by unmodified guests OS
- Works in granularity of blocks (e.g., 2 MiB)

Three main properties per device

- *Size*
  - How much memory is currently plugged
- *Maximum size*
  - How much memory could be plugged
- *Requested size*
  - Request to guest driver to hot(un)plugs blocks to reach requested size

Red Hat

THE LINUX FOUNDATION

virtio-mem: Overview (2)

1. **Prepare for memory devices (here: 16 GB)**

   “/usr/libexec/qemu-kvm -m 4G,maxmem=20G ...”

2. **Create 1..x memory backends, specifying the maximum size**

   “... -object memory-backend-ram,id=mem0,size=16G …”

3. **Create 1..x virtio-mem devices, connecting a backend (optionally specifying a node)**

   “... -device virtio-mem-pci,id=vm0,memdev=mem0,node=0 …”

4. **Request a resize (here: 4 GB)**

   HMP: *“qom-set vm0 requested-size 4G”*

5. **Query current size**

   HMP: *“qom-get vm0 size”*

   HMP: *“info memory-devices”*

```
Memory device [virtio-mem]: "vm0"
  memaddr: 0x100000000
  node: 0
  requested-size: 4294967296
  size: 4294967296
  max-size: 17179869184
  block-size: 2097152
  memdev: /objects/mem0
```

Red Hat

THE LINUX FOUNDATION

virtio-mem: (Dis)Advantages and Open Items

Advantages

- Resize a VM in (configurable) increments - e.g., >= 4 MB on x86-64
- Significantly more flexible than DIMMs and memory ballooning
- Manages VM size changes/requests completely in QEMU (no DIMMs)
- Architecture-independent (e.g., no ACPI)

Disadvantages

- *Not production ready yet* - basic versions are upstream in Linux/QEMU/cloud-hypervisor
- Slower than memory ballooning, cannot “unplug” as much as memory ballooning
- Incompatible with hibernation/suspend

Open Items

- Support other architectures (e.g., arm64, ppc64, s390x) and guest OSs (e.g., Windows)
- *Linux driver: e.g.,* memory hotunplug improvements (WIP), ...
- *QEMU:* e.g., vfio support (WIP), ...
- Libvirt integration
- ...

Summary + Outlook

We now have specialized mechanisms to manage guest memory

- *virtio-balloon:* better interfaces to speed up migration and optimize memory overcommit
- *virtio-pmem:* move pagecache management to the hypervisor
- *virtio-mem:* fine-grained, NUMA-aware memory hot(un)plug

Traditional balloon inflation/deflation remains important

- New mechanisms still have to mature
- Require deeper MM integration - e.g., Windows support difficult

There is still a lot to optimize

- Guest pagecache remains challenging (e.g., virtio-pmem isn't always applicable)
- Encrypted VMs remain challenging
  - The hypervisor isn't allowed to modify (e.g., discard) VM memory content
  - Basic virtio-balloon inflation/deflation and virtio-mem might be feasible. virtio-pmem?
- vfio remains challenging
  - Pins all guest memory, forcing it to remain in hypervisor memory ...
  - At least virtio-mem should be feasible. virtio-balloon? virtio-pmem?


KVM
FORUM

Resources

virtio-spec:

- v1.1: http://docs.oasis-open.org/virtio/virtio/v1.1/virtio-v1.1.pdf
- Latest draft: https://github.com/oasis-tcs/virtio-spec

virtio-balloon

- Free page reporting: https://lore.kernel.org/lkml/20200211224416.29318.44077.stgit@localhost.localdomain/
- Free page hinting:
- https://lore.kernel.org/kvm/1535333539-32420-1-git-send-email-wei.w.wang@intel.com/

virtio-pmem

- QEMU documentation: https://github.com/qemu/qemu/blob/master/docs/virtio-pmem.rst
- Early proposal: https://www.spinics.net/lists/kvm/msg149761.html
- Early discussion: https://www.spinics.net/lists/kvm/msg153095.html

virtio-mem

- Status page: https://virtio-mem.gitlab.io