January 2022

Dynamic Memory Allocation for a Guest Virtual Machine

Udam Saini

Michael Hoyle

Noah Gold

Dynamic Memory Allocation for a Guest Virtual Machine

ABSTRACT

When a virtual machine (VM) runs in the background in an idle state, its memory remains allocated and is unavailable for other host processes or other VMs on the host. This disclosure describes hypervisor-aware virtio ballooning in dynamic host-guest memory allocation. Per the techniques, memory can be dynamically transferred between the guest virtual machines and their host. The inflation operation that enables dynamic transfer of guest VM memory to the host notifies the hypervisor of pages to be freed and also requests the host kernel to free those pages. The techniques can function even in operating systems that lack a way for a hypervisor to subscribe to notifications of page addition or removal events.

KEYWORDS

- Virtual machine (VM)
- Dynamic memory allocation
- Memory ballooning
- Hypervisor
- Memory page
- Virtio
- Virtio balloon
- Paravirtualization
- Hypervisor-aware ballooning
- Virtual machine manager (VMM)

BACKGROUND

A virtual machine (VM) can consume varying amounts of resources. For example, if the VM is hosting a video game, its consumption of computational resources is high. On the other hand, a VM can sometimes run in the background in an idle state when awaiting tasks, without actively consuming memory or other computational resources. Even when a VM is in idle state, the VM memory remains allocated, thereby being unavailable for other host processes or other VMs on the host.

Virtio is a virtualization standard for device drivers, e.g., network, disk, etc. The standard enables the drivers of a guest VM to determine that they are executing in a virtual environment and cooperate with the hypervisor. A virtio-balloon is a paravirtualized device on Unix that enables a host to reclaim memory from the guest, thus reducing memory consumption. A virtio-balloon can be used to reduce the footprint of a VM in idle condition.

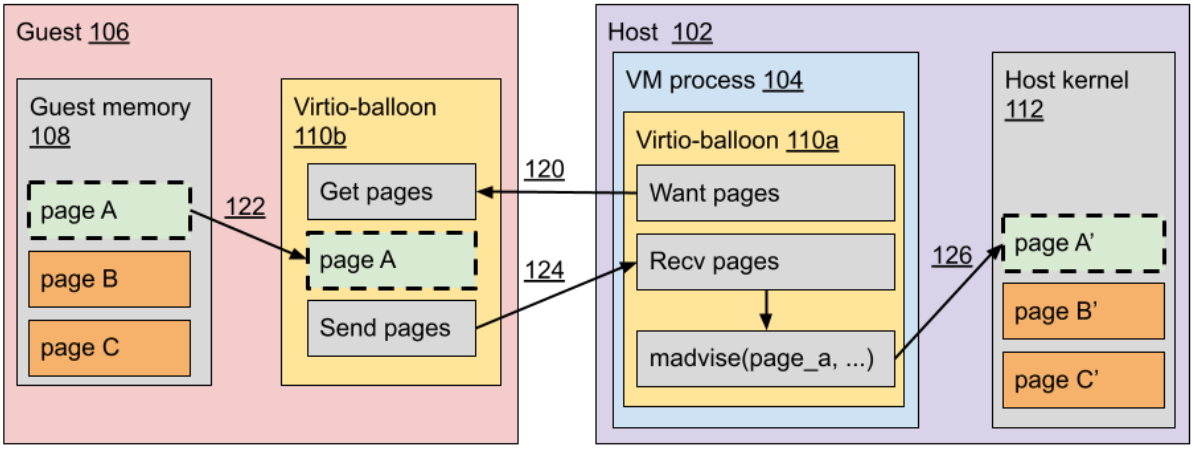

Fig. 1: Transferring memory from a guest VM to its host using virtio-balloons

Fig. 1 illustrates the transfer of memory from a guest VM (106) to its host (102) using virtio-balloons (110a-b). A VM process (104) on the host sends (120) a want-page request to the guest VM. The guest VM transfers (122) from its memory (108) unused pages (page A) to inflate

its virtio-balloon. The guest virtio-balloon (110b) sends (124) the addresses of the unused pages to host virtio-balloon (110a), which receives the pages and returns (126) them to the host kernel (112). The host can then reallocate the pages to other VMs or to other host processes. On Unix-like operating systems, the host uses `madvise()` to tell the kernel that the pages are not in use and that their contents can be discarded. Here, inflate refers to the host reclaiming memory from the guest, while deflate refers to the host returning memory to the guest.

The hypervisor sets up nested page tables, known as two-dimensional page tables (TDP), for the guest, which map from guest virtual address (GVA) to host page frame number (PFN). For the map to remain valid, the host pages must either be locked/pinned, or the hypervisor must be notified when a page is removed. Using `madvise()` or other functions to implement virtio-balloon necessitates a notification mechanism, since the hypervisor must be aware that the PFNs backing parts of the GVA space are no longer available.

In operating systems that lack a way for a hypervisor to subscribe to notifications of such page addition or removal events, the unnotified hypervisor can enable the guest VM to have access to an arbitrary PFN. This risks exposure of data from other host processes and losing contents originally expected to be at that location.

DESCRIPTION

This disclosure describes hypervisor-aware virtio ballooning in dynamic host-guest memory allocation. Per the techniques, the inflation operation that enables dynamic transfer of guest VM memory to the host notifies the hypervisor of pages to be freed and also requests the host kernel to free those pages. The techniques function even in operating systems that lack a way for a hypervisor to subscribe to notifications of page addition or removal events.

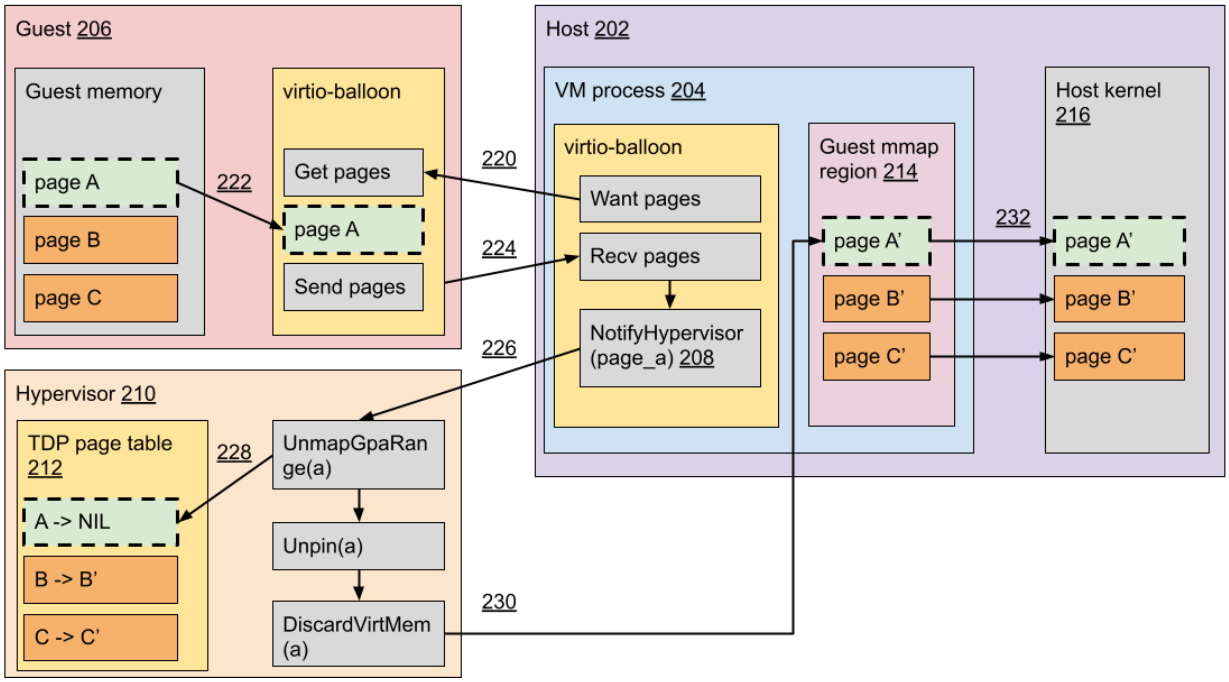

Fig. 2: Hypervisor aware dynamic memory allocation between the host and a guest VM

Fig. 2 illustrates hypervisor-aware dynamic memory allocation between the host and the guest VM. A VM process (204) on the host (202) sends (220) a want-page request to the guest VM (206). The guest VM transfers (222) unused pages (page A) from its memory to inflate its virtio-balloon. The guest virtio-balloon sends (224) the addresses of the unused pages to the host virtio-balloon.

Upon receipt of unused page addresses, the host notifies (226)the hypervisor (210) using, e.g., a `NotifyHypervisor()` or similar command (208). The hypervisor uses a series of instructions, e.g., `UnmapGpaRange()`, `Unpin()`, `DiscardVirtMem()`, etc., to update (228) its two-dimensional page table (212) for the guest. The host receives (230) a signal from the hypervisor that the de-allocated page (page A) has been updated in its TDP table, and the host removes (232) the address from its guest memory-map region (214) and returns it to the host kernel (216).

Per the techniques, the hypervisor exposes a way to correctly and safely punch holes (inflate the balloon) in guest memory. Punching a hole H causes the following actions:

- Clears the TDP entries for H in the physical address space of the guest.
- Remove H from the working set of the virtual machine manager (VMM).
- Enables the VMM/hypervisor to repopulate the guest page frames (GFNs) in H when the balloon is deflated.

Repopulating GFNs (deflating the balloon) includes the following.

- Ensuring that the memory-mapped guest memory region created by the VM can safely accept reads and writes to the previously dropped pages.
- Creating a new TDP mapping for the previously ballooned GFNs pointing at the potentially new PFNs backing the part of the guest memory map being repopulated.

In this manner, the techniques of this disclosure enable optimized usage of memory and other resources when a guest virtual machine is in an idle state, even in operating systems that lack a way for the hypervisor to subscribe to notifications of page addition or removal events. The host system can reclaim memory from the VM by inflating the memory balloon inside the VM, which reduces the memory available to other tasks inside the VM. The guest OS decides the

memory pages to be given back to the host by marking them as unavailable for the guest VM. Effectively, the virtio-balloon driver holds on to those PFNs such that they cannot be used by anything else in the guest. Since the virtio-balloon is holding those pages, it is safe to return them to the host. The host can then use these for other processes or VMs. If at a later time the VM requests more memory, the host can return pages to the guest and shrink the holes. This enables dynamic adjustment of the memory available to a VM even while the VM is running.

CONCLUSION

This disclosure describes hypervisor-aware virtio ballooning in dynamic host-guest memory allocation. Per the techniques, memory can be dynamically transferred between the guest virtual machines and their host. The inflation operation that enables dynamic transfer of guest VM memory to the host notifies the hypervisor of pages to be freed and also requests the host kernel to free those pages. The techniques can function even in operating systems that lack a way for a hypervisor to subscribe to notifications of page addition or removal events.

REFERENCES

[1] Phillipp Hahn, “VirtIO Memory Ballooning,” https://pmhahn.github.io/virtio-balloon/ accessed December 26, 2021.

[2] “madvise(2) - Linux manual page” https://man7.org/linux/man-pages/man2/madvise.2.html accessed December 26, 2021.