The challenges of asynchronous teardown

Claudio Imbrenda

IBM

12 September 2022

Disclaimers

Overview

Introduction and motivation

When a very large process terminates, it can take a considerable time before all the used memory is freed.

This is especially true for large (multi-TB) protected VMs, for example using IBM Secure Execution for Linux on s390x hosts.

Rebooting a protected VM on s390x also poses additional challenges.

The goal is to allow QEMU to terminate immediately and leave the teardown of the memory to an asynchronous process.

An example

- On my workstation (`Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz`)
- Simple process that allocates and uses a lot of memory, then exits.
- Tested from 4GB to 28GB in increments of 4GB.
- Teardown speed uniform at ~13GB/s for all sizes.

Actual results		Extrapolated	
Size	Time	Size	Time
4GB	0.3s	64GB	4.9s
8GB	0.6s	128GB	9.8s
12GB	0.9s	256GB	19.7s
16GB	1.3s	1TB	78.8s
20GB	1.6s	4TB	315.1s
24GB	2.0s	8TB	630.2s
28GB	2.2s	16TB	1260.3s

Potential issues with asynchronous teardown

- Resource allocation issues
- OOM killer
- Proper accounting
- Implementation complexity

Reboot vs shutdown

Rebooting a protected VM on s390x means effectively to destroy and clear all of the guest memory, and then set up a new protected guest after the reboot. Guest memory and userspace process are still present.

In the shutdown case, instead, the memory is already gone by the time KVM gets in control.

Reboot – overview of possible solutions

Problem – so far – only relevant on s390x.

- Kernel thread
- Userspace thread

Reboot – kernel thread

- When destroying a protected VM, start a kernel thread to clean up the guest memory
- Allow to concurrently start a new (normal or protected) VM in the same memory

Reboot – kernel thread

- When destroying a protected VM, start a kernel thread to clean up the guest memory
- Allow to concurrently start a new (normal or protected) VM in the same memory

Advantages

- No userspace changes needed
- No common code changes

Disadvantages

- Improper accounting of CPU time (cgroups)

Reboot – userspace thread

- When destroying a protected VM, userspace should indicate through a new interface that the protected guest is to be destroyed asynchronously.
- Allow to concurrently start a new (normal or protected) VM in the same memory
- Userspace triggers the teardown in a separate thread

Reboot – userspace thread

- When destroying a protected VM, userspace should indicate through a new interface that the protected guest is to be destroyed asynchronously.
- Allow to concurrently start a new (normal or protected) VM in the same memory
- Userspace triggers the teardown in a separate thread

Advantages

- Proper accounting of CPU time (QEMU, cgroups)
- No common code changes

Disadvantages

- Requires userpace changes

Reboot – outcome

Proper accounting was deemed worth the effort of requiring userspace changes.

Status:

Kernel:	last patches to be merged, hopefully in 6.1
QEMU:	tbd
libvirt:	no changes needed

`https://lore.kernel.org/lkml/20220810125625.45295-1-imbrenda@linux.ibm.com/`

Shutdown – overview of possible solutions

- Kernel thread
- User thread
- Deferred mmput
- Clone(CLONE_VM)

Shutdown – kernel thread

- When freeing guest memory, use `get_page` to pin each page
- Put the pinned pages in a list
- When KVM is torn down, start a new kernel thread
- The kernel thread will process and free all pages in the list

Shutdown – kernel thread

Advantages

- No userspace changes
- Arch-specific

Disadvantages

- Improper accounting of CPU time (cgroups)
- Large impact on memory management (pinned memory cannot be swapped or reclaimed)
- Complex interaction with the oom killer
- Complex implementation
- Arch-specific

Shutdown – user thread

- Before shutdown (or even at process startup), fork/clone a new “cleaner” process
- The new process will call an IOCTL, which will sleep
    - the IOCTL will sleep while in the kernel
    - uninterruptible, unkillable
- When freeing guest memory, use `get_page` to pin each page
- Put the pinned pages in a list
- When KVM is torn down, wake the user process that was sleeping in the IOCTL
- The IOTCL will process and free all pages in the list
    - running in the context of the cleaner process
    - still unkillable until done

Shutdown – user thread

Advantages

- Proper accounting of CPU time (cgroups)
- Arch-specific

Disadvantages

- Large impact on memory management (pinned memory cannot be swapped or reclaimed)
- Userspace changes (common code and/or arch code)
- Complex interaction with the oom killer
- Complex implementation
- Arch-specific

Shutdown – Deferred mmput

- Arch-specific hooks mark the mm (e.g. for protected VMs)
- Use `mmput_async` instead of `mmput` when tearing down marked mms
- The teardown will then happen in a kernel worker thread.

Shutdown – Deferred mmput

- Arch-specific hooks mark the mm (e.g. for protected VMs)
- Use `mmput_async` instead of `mmput` when tearing down marked mms
- The teardown will then happen in a kernel worker thread.

Advantages

- Simple implementation
- Architecture independent
- No userspace changes needed

Disadvantages

- Improper accounting of CPU time (cgroups)
- Common code changes

Shutdown – `clone(CLONE_VM)`

- Before shutdown (ideally at startup), a second “cleaner” process is cloned
  - using `clone(CLONE_VM)`
  - the new process will share the address space of the parent, without being a thread
- When the parent terminates, no memory cleanup is performed
  - because the mm is still used by the child process
- The cleaner process will wait until the parent process has completely terminated
- The cleaner process can simply exit

Shutdown – `clone(CLONE_VM)`

Advantages

- Proper accounting of CPU time (cgroups)
- Simple implementation (~200 lines of code including comments)
- Architecture independent
- Completely in userspace

Disadvantages

- Userspace changes (common code)
- The clean-up process is killable

Shutdown – outcome

The shutdown part is still under discussion upstream, but a consensus seems to have formed around the last solution.

Status:

Kernel:	no changes needed
QEMU:	tbd
libvirt:	no changes needed

`https://lore.kernel.org/all/20220812133453.82671-1-imbrenda@linux.ibm.com/`

Technical details

- Command line option to enable deferred teardown
    - opt-in feature
- At QEMU startup create a new “cleaner” process with `clone` and `CLONE_VM`
    - the new process is not a thread
- The cleaner process calls `prctl(PR_SET_PDEATHSIG, SIGHUP)`
    - the kernel will send the specified signal to the cleaner process when its parent exits
- The cleaner process also closes all open file descriptors
    - If available, using `close_range`
    - Otherwise looking for the open file descriptors in `/proc/self/fd`
    - This is also needed to make libvirt happy
- The cleaner process waits for the signal
- Once the parent process has terminated completely, the cleaner process can exit

Final remarks

- Generalized solution for asynchronous teardown of any process
- Should be spun off into a library, so other heavy processes (e.g. databases) can use the same mechanism?

Questions?