Linux Plumbers Conference

Dublin, Ireland September 12-14, 2022

CRIU

Virtuozzo OpenVZ

Bringing up FUSE mounts C/R support

How CRIU handles filesystems?

- we have a special structure called fstype:
  - name / code
  - dump
  - restore
  - parse
  - can_mount
  - sb_equal

Example: tmpfs

- tmpfs_dump
  - save the entire filesystem to the image file (tar.gz archive)
- tmpfs_restore
  - extracts the entire tree after new mount is ready

Block-device based: ext4, xfs, …

- we don’t handle them as:
  - in most scenarios this filesystems is an external mounts
  - can only be mounted by the root user (only one such mount in container and bindmounts)

Example: overlayfs

- mount
  - parses option string, resolves paths and build a new mount option string
- can_mount
  - checks that all mount dependencies are met
    - lowerdirs
    - upperdir
    - workdir

NFS (OpenVZ fork)

- We can’t mount NFS (no network)
- Mount Stub-Proxy File System (SPFS) with two modes:
  - stub - hangs on any IO
  - proxy - translates all fs actions to some directory
- Set “proxy” mode
- Perform files open (restore stage)
- Set “stub” mode
- After CRIU restore finishes…
- freeze again and do spfs lazy umount, mount nfs on that place
- iterate over processes and replace file descriptors on the fly

When are mounts restored?

1. mounts (prepare_mnt_ns())
2. tasks
   a. mappings (prepare_mappings() -> premap_priv_vmas())

FUSE challenges

1. FUSE daemon is required to mount
2. FUSE daemon is the process
   - it may have shared memory
   - it may have network sockets

=> we can’t restore it separately from pstree (or just run a new daemon without saving the state)
=> it’s impossible to restore fuse mounts? Hope that it’s possible. :-)

Problem 1: mounting

- can’t move fuse mount timeslot
- let’s spawn a fake daemon and perform mounting
- .. then replace the fake daemon with the original one (once the process get ready!)

Fuse mounting

1. open /dev/fuse device
2. build mount options string with the “fd” parameter
3. mount
   a. fills struct file -> private_data
      refer to fuse_dev_alloc_install & fuse_fill_super_common
4. answer to FUSE_INIT request
   a. fuse_send_init
5. do read/write on this fd
   a. fuse_dev_read / fuse_dev_write

Daemon replacement

- spawn “fake daemon” process
- open /dev/fuse
- perform mounting
- save /dev/fuse file descriptor in CRIU fdstore
- once real fuse daemon is ready, kill fake and send /dev/fuse control fd to the real one
- That’s it!

Problem 2: opening files

- fake daemon is able to process only FUSE_INIT req
- what about fuse file descriptors C/R?

CRIU: how are files restored?

- we have file_desc_ops (restore) and fdtype_ops (dump)
  - each task -> prepare_fds -> open_fdinfos
- file_desc_ops have to provide .open callback
- … but we can't open because of the fake daemon!
- => we need to extend daemon capabilities to allow opening files "a dumb" way
- => possibly, we'll need some kernel modification here

fuse_inode id

- struct fuse_inode -> nodeid - unique identifier between userspace / kernel
- userspace daemon uses it to distinguish the files
  - so, this nodeid keeps in daemon process memory
  - => we need to restore keep it
  - fortunately, userspace may control it (FUSE_LOOKUP response)
- We also need to dump this nodeid
  - use fuse fhandles
    - at least fuse_encode_fh provides it

What about FUSE file mappings?

- mmap doesn’t lead to fuse request
  - just generic filemap_fault is used
  - fuse_file_apos (struct address_space_operations)
    - readpage callback leads to simple fuse READ request and filling page cache
- DAX is not covered here at all!

What about dump stage?

- CRIU freezes all processes (they are under ptrace)
- => fuse daemon gets frozen too
- => any I/O request to fuse mount leads to D-state
- Only “stat” syscall is needed on the dump stage
  - … for non-ghost and regular files
- We need something like a “pre-dump” stage but for files
  - make stat before freezing and save info
  - check that fd is the same on the dump (kcmp syscall extension?…)

Status & Plans

- we’ve PoC of FUSE daemon replacement
- write the initial implementation for minimalistic fuse fs C/R
- try to cover more complex cases like network-based filesystems
- … fusectl support? (opened files from it!)
- … fuseblk?
- … cuse?
- different fuse versions (ABIs) from the kernel side?

References

[1] CRIU github.com/checkpoint-restore/criu
[2] github.com/libfuse/libfuse
[3] Linux kernel fs/fuse

Thanks!

Linux
Plumbers
Conference

Dublin, Ireland September 12-14, 2022

CRIU

Virtuozzo

OpenVZ