Checkpoint-restoring containers with Docker inside

Alexander Mikhalitsyn
Software Developer at Virtuozzo

Pavel Tikhomirov
Software developer at Virtuozzo

Aug 2020
Agenda

- Nested PID namespaces
- Several UTS/IPC namespaces
- Nested USER namespaces
- Internal overlayfs mounts
Namespaces are stored in global single linked list ns_ids (struct ns_id).

Each ns_id has struct ns_desc - namespace descriptor (clone flag, “string” name in /proc/pid/ns/<..>)

nsid_add() function adds new namespace to list

rst_add_ns_id() is wrapper around nsid_add() to add new ns_id on restore stage

__get_ns_id() is wrapper of nsid_add() on dump stage
Virtuozzo

General namespace C/R design (OLD)

**Dump**

- cr_dump_tasks
  - collect_pstree_ids
  - loop get_task_ids
  - dump_task_ns_ids
  - __get_ns_id
- We have no separate image for ns_id info

**Restore**

- cr_restore_tasks
  - prepare_pstree
  - read_pstree_image
  - loop read_one_pstree_item
  - read_pstree_ids
  - rst_add_ns_id
- Entering to namespace in restore_one_alive_task or earlier
**Virtuozzo**

General namespace C/R design (NEW)

**Dump**

- `cr_dump_tasks`
  - `collect_pstree_ids`
  - `loop get_task_ids`
  - `dump_task_ns_ids`
  - `__get_ns_id`
- We have separate image and can store parent nsid, users id.
- `collect_ns_hierarhy`
  - `loop set_ns_hookups`
  - `set_ns_opt (parent, users)`

**Restore**

- `cr_restore_tasks`
  - `read_ns_with_hookups`
  - `rst_new_ns_id`
- Entering to namespace in `restore_one_alive_task` or earlier
● We can do setns and go to the USER namespace immediately
● USER namespaces can be nested
● So, we can reconstruct hierarchy of user namespaces independently from processes tree and then do setns() on each process
● But we can’t do setns to higher USER ns from lower-level
Virtuozzo

Nested USER namespaces

Dump

- `cr_dump_tasks`
  - `collect_pstree_ids`
  - `loop get_task_ids`
  - `dump_task_ns_ids`
  - `__get_ns_id`

- `collect_user_namespaces`
  - `loop collect_user_ns`
  - `__dump_user_ns`

- `collect_ns_hierarhy`
  - `loop set_ns_hookups`
  - `set_ns_opt (parent, users)`

Restore

- `prepare_namespace`
  - `create_user_ns_hierarhy`
  - `prepare_usersns`

- `restore_one_alive_task`
  - `set_user_ns`
  - `setns`
• We can’t do setns and go to the PID namespace immediately
• PID namespaces can be nested
• So during restoring pstree we call clone with CLONE_NEWPID flag to create and get into correct pidns from the birth of the process
• Will call setns to restore pidns for children
Dump

- `cr_dump_tasks`  
  -> `collect_pstree_ids`  
    -> `loop get_task_ids`  
    -> `dump_task_ns_ids`  
    -> `__get_ns_id`

- `collect_ns_hierarhy`  
  -> `loop set_ns_hookups`  
  -> `set_ns_opt (parent, users)`

Restore

- `restore_task_with_children`  
  -> `setup_current_pid_ns`  
  -> `create_pid_ns_helper`

- `restore_one_alive_task`  
  `restore_task_pfc_before_user_ns`  
  `set_pid_for_children_ns`

- `restore_one_alive_task`  
  `sigreturn_restore`  
  `__export.restore_task`  
  `__export.restore_thread`  
  `setns(CLONE_NEWPID)`

- `destroy_pid_ns_helpers`  
  -> `loop SIGKILL`
• Process groups (we can setpgid(pid) - \textbf{ok})
• Sessions (inherit) (we can only create new session and inherit - \textbf{problem})
• Sessions restore when some tasks were reparented to init
• Process trees where some process (in one session) has set \texttt{PR\_SET\_CHILD\_SUBREAPER} and reparenting occurred for the process from another session
Virtuozzo

Nested PID namespaces (sessions 1)

1. dies

2. reparenting to init
Nested PID namespaces (sessions 2)

1. init (pid=1, sid=1)
2. A (pid=2, sid=1)
3. B (pid=3, sid=3)
4. D (pid=5, sid=3)
5. 3. Can't inherit session
Virtuozzo

Several UTS/IPC namespaces

Dump

- collect_pstree_ids
  -> for_each_pstree_item
  -> dump_task_ns_ids

  ... dump_pstree(root_task)...

- dump_namespaces(root_task)
  -> dump_uts_ns (or ipc, or net)

Restore

- restore_task_with_children
  -> make_root_ns(uts)

- restore_task_with_children
  -> prepare_namespace(root_task)
    -> prepare_namespaces(uts)

- restore_one_alive_task
  -> restore_task_ns (do setns)
● Overlayfs mounts have several path options: lowerdir, workdir, upperdir

● Example:
  overlay on /var/lib/docker/overlay2/XYZ/merged type overlay

● But also overlayfs could be mounted with relative paths in these options
After pivot_root syscall paths won’t be recalculated!
overlay on /var/lib/docker/overlay2/XYZ/merged type overlay
(<...>,lowerdir=/tmp/mntns.XXXX./yard/var/lib/docker/overlay2/XYZ-init
diff:/tmp/mntns.XXXX./yard/var/lib/docker/overlay2/
ABC/diff,upperdir=/tmp/mntns.XXXX./yard/var/lib/docker/overlay2/
XYZ/diff,workdir=/tmp/mntns.XXXX./yard/var/lib/docker/
overlay2/XYZ/work)

We don’t know from which mount these paths were opened
2. Overlayfs support in CRIU [https://src.openvz.org/projects/OVZ/repos/criu/commits/8ee9695522f4f98a21ef7fad23da9ccc72ecdc6](https://src.openvz.org/projects/OVZ/repos/criu/commits/8ee9695522f4f98a21ef7fad23da9ccc72ecdc6)
3. Namespaces connections ioctls [https://lwn.net/Articles/698364/](https://lwn.net/Articles/698364/)
4. UTS/IPC namespaces [https://src.openvz.org/projects/OVZ/repos/criu/commits/705e08f9d8bb40cf455296bcb987d2f26cd39401](https://src.openvz.org/projects/OVZ/repos/criu/commits/705e08f9d8bb40cf455296bcb987d2f26cd39401)
5. Overlayfs kernel patch [https://lore.kernel.org/lkml/20200604161133.20949-1-alexander.mikhalitsyn@virtuozzo.com/](https://lore.kernel.org/lkml/20200604161133.20949-1-alexander.mikhalitsyn@virtuozzo.com/)
6. nsfs: Introduce ioctl to set vector of ns_last_pid's on pid ns hierarhy [https://lore.kernel.org/lkml/149245014695.17600.12640895883798122726.stgit@localhost.localdomain/](https://lore.kernel.org/lkml/149245014695.17600.12640895883798122726.stgit@localhost.localdomain/)

Links
Thank you!