Subject: [PATCH 07/11] restart: explicitly disallow orphan tasks with --no-pidns
Posted by Oren Laadan on Mon, 07 Feb 2011 17:21:28 GMT
View Forum Message <> Reply to Message

Add a test to ckpt_build_tree() that, if --no-pidns is required, will
verify that the tasks data structure will not lead to an orphan task
during the restart (since this should not have been possible in the
original subtree checkpoint).

This is useful beyond a sanity check:

If --no-pidns is set, then every child task asks to be notified of the
parent's death via prctl (for better cleanup). This is ok because from
a subtree checkpoint there should never be orphan tasks. But if we do
have an orphan task (due to e.g. a bug or incorrectly modified image),
then the following may happen:

A ghost/dead task that has a child (should be impossilbe in subtree
image) may exit before that child; In that case, the child receives
the signal and dies prematurely. Restart may either fail if this
happens early enough, or simply hang since the coordinator will be
waiting forever for all the tasks to complete.

Besides the sanity test, we also modify the parent death signal to be
SIGUSR1 instead of SIGKILL, and we catch it - so if the race happens
while the restarting task is still in userspace, we can be verbose
about it. We also add a hanlder for SIGSEGV, so we'll get a message
about it too.

Signed-off-by: Oren Laadan <orenl@cs.columbia.edu>
---
 restart.c | 50 ++-----
 1 files changed, 45 insertions(+), 5 deletions(-)

diff --git a/restart.c b/restart.c
index 966f7a1..01566c2 100644
--- a/restart.c
+++ b/restart.c
@@ -306,6 +306,18 @@ static char *sig2str(int sig)
  return "UNKNOWN SIGNAL";
 }

+static void sigusr1_handler(int sig)
+{
+ ckpt_dbg("SIGUSER1: restarting process would become orphan.\n");
+ exit(1);
+}
+

```
+static void sigsegv_handler(int sig)
+{
+ ckpt_dbg("SIGSEGV: restarting process unexpected fault.\n");
+ exit(1);
+}
+
 static void sigchld_handler(int sig)
 {
  int collected = 0;
@@ -573,6 +585,7 @@ int cr_restart(struct cr_restart_args *args)
    ckpt_perror("unshare");
    goto cleanup;
   }
+
   /* chroot ? */
   if (args->root && chroot(args->root) < 0) {
    ckpt_perror("chroot");
@@ -912,6 +925,9 @@ static int __ckpt_coordinator(void *arg)

  /* none of this requires cleanup: we're forked ... */

+ /* we want to be vocal about sigsegv */
+ signal(SIGSEGV, sigsegv_handler);
+
  /* chroot ? */
  if (ctx->args->root && chroot(ctx->args->root) < 0) {
   ckpt_perror("chroot");
@@ -1114,7 +1130,7 @@ static inline struct task *ckpt_init_task(struct ckpt_ctx *ctx)
 static int ckpt_build_tree(struct ckpt_ctx *ctx)
 {
  struct task *task;
- int i;
+ int i, found;

  /*
   * Allow for additional tasks to be added on demand for
@@ -1167,6 +1183,25 @@ static int ckpt_build_tree(struct ckpt_ctx *ctx)
  ckpt_dbg(".....\n");
 #endif

+ /*
+  * For --no-pidns verify that the resulting tree won't create
+  * orphan tasks (comment in ckpt_fork_stub() explain why).
+  */
+ if (!ctx->args->pidns) {
+  found = 0;
+  for (i = 0; i < ctx->tasks_nr; i++) {
+   task = &ctx->tasks[i];
```

```
+   if (!(task->flags & (TASK_GHOST | TASK_DEAD)))
+    continue;
+   if (!task->children)
+    continue;
+   ckpt_err("Bad orphan task (#%d) with --no-pidns\n", i);
+   found = 1;
+  }
+  if (found)
+   return ctx_ret_errno(ctx, EINVAL);
+ }
+
  return 0;
 }

@@ -1814,15 +1849,20 @@ int ckpt_fork_stub(void *data)
   * death by asking to be killed then. When restart succeeds,
   * it will have replaced this with the original value.
   *
-  * This works because in the --no-pids case, the hierarchy of
-  * tasks does not contain zombies (else, there must also be a
-  * container init, whose pid (==1) is clearly already taken).
+  * This works because in the --no-pidns case, the hierarchy of
+  * tasks does not contain zombies; else, there must also be a
+  * container init, whose pid (==1) is clearly already taken.
+  * This is verified in ckpt_build_tree().
   *
   * Thus, if a the parent of this task dies before this prctl()
   * call, it suffices to test getppid() == task->parent_pid.
+  *
+  * We use SIGUSR1 so that we can catch it and issue an error
+  * or debug message when this happens.
   */
  if (!ctx->args->pidns) {
-  if (prctl(PR_SET_PDEATHSIG, SIGKILL, 0, 0, 0) < 0) {
+  signal(SIGUSR1, sigusr1_handler);
+  if (prctl(PR_SET_PDEATHSIG, SIGUSR1, 0, 0, 0) < 0) {
    ckpt_perror("prctl");
    return ctx_set_errno(ctx);
   }
--
1.7.1
```