Subject: [PATCH 03/11] restart: make feeder a proper child instead of a thread
Posted by Oren Laadan on Mon, 07 Feb 2011 17:21:24 GMT
View Forum Message <> Reply to Message

As pointed out by Sukadev Bhattiprolu in this post:
http://www.spinics.net/lists/linux-containers/msg22411.html
it's not a good idea to have the coordinator and feeder share the same
memory address space.

The original idea was prevent the feeder from generating a SIGCHLD
prematurely that will interrupt the restart. So we could use regular
clone() without SIGCHLD. But then, if the feeder exits last then it
will spit an aesthetic message _after_ the "succes" message from the
coordiantor.

This patch makes the feeder a proper child, but also makes the feeder
wait for the coordinator before terinating, and makes the coordiantor
collect the feeder.

Cc: Sukadev Bhattiprolu <sukadev@linux.vnet.ibm.com>
Signed-off-by: Oren Laadan <orenl@cs.columbia.edu>

```
---
 restart.c |   86 +++++++++++++++++++++++++++++++++++++++++-------------------------------------
 1 files changed, 49 insertions(+), 37 deletions(-)

diff --git a/restart.c b/restart.c
index 195a892..9535543 100644
--- a/restart.c
+++ b/restart.c
@@ -174,6 +174,7 @@ static int global_ulogfd;
 static int global_uerrfd;
 static int global_debug;
 static int global_verbose;
+static pid_t global_feeder_pid;
 static pid_t global_child_pid;
 static int global_child_status;
 static int global_child_collected;
@@ -205,7 +206,7 @@ static pid_t ckpt_fork_child(struct ckpt_ctx *ctx, struct task *child);
 static int ckpt_adjust_pids(struct ckpt_ctx *ctx);

 static void ckpt_abort(struct ckpt_ctx *ctx, char *str);
-static int ckpt_do_feeder(void *data);
+static int ckpt_do_feeder(struct ckpt_ctx *ctx);
 static int ckpt_fork_feeder(struct ckpt_ctx *ctx);

 static int ckpt_write(int fd, void *buf, int count);
@@ -313,11 +314,17 @@ static void sigchld_handler(int sig)
   ckpt_dbg("SIGCHLD: child not ready\n");
```

```
    break;
   } else if (pid > 0) {
-   /* inform collection coordinator or root-task */
+   /* inform collection of coordinator or root-task */
    if (pid == global_child_pid) {
     global_child_status = status;
     global_child_collected = 1;
-    report_exit_status(status, "SIGCHLD: ", 1);
+    ckpt_dbg("collected coord/root task\n");
+    report_exit_status(status, "SIGCHLD:", 1);
+   }
+   /* collect the feeder child */
+   if (pid == global_feeder_pid) {
+    ckpt_dbg("collected feeder process\n");
+    report_exit_status(status, "SIGCHLD:", 1);
    }
    ckpt_dbg("SIGCHLD: collected child %d\n", pid);
    collected = 1;
@@ -521,7 +528,7 @@ static void exit_ctx(struct ckpt_ctx *ctx)
 int cr_restart(struct cr_restart_args *args)
 {
  struct ckpt_ctx ctx;
- int ret;
+ int status, ret;

  init_ctx(&ctx);

@@ -649,17 +656,30 @@ int cr_restart(struct cr_restart_args *args)
   ret = ckpt_coordinator(&ctx);
  }

- /*
-  * On success, return pid of root of the restart process tree.
-  */
-
  if (ret < 0)
   goto cleanup;

+ /* success: return pid of root of the restart process tree */
  ret = global_child_pid;

+ /* time to release feeder so he can peacefully retire now */
+ status = 0;
+ if (write(ctx.pipe_out, &status, sizeof(status)) != sizeof(status))
+  ret = -1;
+
  cleanup:
  exit_ctx(&ctx);
```

```
+
+ /* feeder doesn't exit - to avoid SIGCHILD to coordinator */
+ if (ret < 0 && global_feeder_pid)
+  kill(global_feeder_pid, SIGKILL);
+ /* wait for feeder child to terminate (ok of already gone) */
+ if (global_feeder_pid)
+  waitpid(global_feeder_pid, NULL, 0);
+
+ if (ret < 0)
+  errno = ctx.error;
+
  return ret;
 }

@@ -1906,38 +1926,30 @@ static pid_t ckpt_fork_child(struct ckpt_ctx *ctx, struct task *child)
  */
 static int ckpt_fork_feeder(struct ckpt_ctx *ctx)
 {
- genstack stk;
  pid_t pid;
+ int ret;

  if (pipe(ctx->pipe_feed)) {
   ckpt_perror("pipe");
-  return -1;
+  return ctx_set_errno(ctx);
  }

  if (pipe(ctx->pipe_child) < 0) {
   ckpt_perror("pipe");
-  return -1;
- }
-
- /*
-  * Use clone() without SIGCHLD so that the when the feeder
-  * terminates it does not notify the parent (coordinator), as
-  * this may interfere with the restart.
-  */
-
- stk = genstack_alloc(PTHREAD_STACK_MIN);
- if (!stk) {
-  ckpt_perror("ckpt_fork_feeder genstack_alloc");
-  return -1;
+  return ctx_set_errno(ctx);
  }

- pid = clone(ckpt_do_feeder, genstack_sp(stk),
-     CLONE_THREAD | CLONE_SIGHAND | CLONE_VM, ctx);
```

```
+ pid = fork();
  if (pid < 0) {
   ckpt_perror("feeder thread");
-  return -1;
+  return ctx_set_errno(ctx);
+ } else if (pid == 0) {
+  ret = ckpt_do_feeder(ctx);
+  exit(ret);
  }

+ global_feeder_pid = pid;
+
  /* children pipe: used for status reports from children */
  close(ctx->pipe_child[0]);
  ctx->pipe_out = ctx->pipe_child[1];
@@ -2045,12 +2057,15 @@ ckpt_dbg("write len %d (%d)\n", len, ret);
  * In '--no-pids' mode, transform the pids array (struct ckpt_pids)
  * on the fly and feed the result to the "init" task of the restart
  */
-static int ckpt_do_feeder(void *data)
+static int ckpt_do_feeder(struct ckpt_ctx *ctx)
 {
- struct ckpt_ctx *ctx = (struct ckpt_ctx *) data;
+ int status;

  ctx->whoami = CTX_FEEDER;  /* for sanity checks */

+ if (prctl(PR_SET_PDEATHSIG, SIGKILL, 0, 0, 0) < 0)
+  ckpt_abort(ctx, "prctl");
+
  /*
   * feeder has a separate file descriptor table, so
   * close/dup/open etc do not affect original caller
@@ -2096,16 +2111,13 @@ static int ckpt_do_feeder(void *data)
  else
   ckpt_read_write_blind(ctx);

- /* All is well: feeder thread is done.  However, we must
-  * invoke the exit system call directly. Otherwise, upon
-  * return from this function, glibc's clone wrapper will call
-  * _exit, which calls exit_group, which will terminate the
-  * whole process, which is not what we want.
-  */
- syscall(SYS_exit, 0);
+ /* wait for parent (coordinator) to confirm, to avoid
+    prematurely interrupting the restart with SIGCHLD */
+ if (read(ctx->pipe_in, &status, sizeof(status)) != sizeof(status))
+  ckpt_abort(ctx, "read coord status");
```

```
- /* not reached */
- return 0;
+ close(ctx->pipe_in);  /* no need to mark unused */
+ return status;
 }

 /*
--
1.7.1
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs