Subject: [PATCH 06/11] restart: rename 'ctx->tasks_arr' to 'ctx->tasks'
Posted by Oren Laadan on Mon, 07 Feb 2011 17:21:27 GMT
View Forum Message <> Reply to Message

In preparation for next (super)-patch.

Signed-off-by: Oren Laadan <orenl@cs.columbia.edu>
---
 restart.c | 44 ++++++++++++++++++++++++++----------------
 1 files changed, 22 insertions(+), 22 deletions(-)

```
diff --git a/restart.c b/restart.c
index 05d101b..966f7a1 100644
--- a/restart.c
+++ b/restart.c
@@ -141,7 +141,7 @@ struct ckpt_ctx {
  struct ckpt_pids *copy_arr;
  __s32 *vpids_arr;

- struct task *tasks_arr;
+ struct task *tasks;
  int tasks_nr;
  int tasks_max;
  int tasks_pid;
@@ -521,8 +521,8 @@ static void exit_ctx(struct ckpt_ctx *ctx)
 {
  if (ctx->freezer)
   free(ctx->freezer);
- if (ctx->tasks_arr)
-  free(ctx->tasks_arr);
+ if (ctx->tasks)
+  free(ctx->tasks);
  if (ctx->pids_arr)
   free(ctx->pids_arr);
  if (ctx->copy_arr)
@@ -641,13 +641,13 @@ int cr_restart(struct cr_restart_args *args)
   * setup devpts, root-dir and /proc if necessary, ...
   */
  if (ctx.args->mnt_pty)
-  ctx.tasks_arr[0].flags |= TASK_NEWPTS;
+  ctx.tasks[0].flags |= TASK_NEWPTS;
  if (ctx.args->mntns)
-  ctx.tasks_arr[0].flags |= TASK_NEWNS;
+  ctx.tasks[0].flags |= TASK_NEWNS;
  if (ctx.args->root)
-  ctx.tasks_arr[0].flags |= TASK_NEWROOT;
+  ctx.tasks[0].flags |= TASK_NEWROOT;
```

```
- if (ctx.args->pidns && ctx.tasks_arr[0].pid != 1) {
+ if (ctx.args->pidns && ctx.tasks[0].pid != 1) {
   ckpt_dbg("new pidns without init\n");
   if (global_send_sigint == -1)
    global_send_sigint = SIGINT;
@@ -656,12 +656,12 @@ int cr_restart(struct cr_restart_args *args)
    * the coordinator should set up the filesystems and
    * not the first process in the application process tree.
    */
-  ctx.tasks_arr[0].flags &=
-   ~(TASK_NEWPTS | TASK_NEWROOT |TASK_NEWNS);
+  ctx.tasks[0].flags &=
+   ~(TASK_NEWPTS | TASK_NEWROOT | TASK_NEWNS);
   ret = ckpt_coordinator_pidns(&ctx);
  } else if (ctx.args->pidns) {
   ckpt_dbg("new pidns with init\n");
-  ctx.tasks_arr[0].flags |= TASK_NEWPID | TASK_NEWNS;
+  ctx.tasks[0].flags |= TASK_NEWPID | TASK_NEWNS;
   if (global_send_sigint == -1)
    global_send_sigint = SIGKILL;
   ret = ckpt_coordinator(&ctx);
@@ -1039,7 +1039,7 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)
  pid_t root_pid;
  int ret;

- root_pid = ckpt_fork_child(ctx, &ctx->tasks_arr[0]);
+ root_pid = ckpt_fork_child(ctx, &ctx->tasks[0]);
  if (root_pid < 0)
   return -1;
  global_child_pid = root_pid;
@@ -1069,7 +1069,7 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)
   ret = 0;
  }

- if (ctx->args->pidns && ctx->tasks_arr[0].pid != 1) {
+ if (ctx->args->pidns && ctx->tasks[0].pid != 1) {
   /* Report success/failure to the parent */
   if (ret < 0)
    ret = ctx->error;
@@ -1104,7 +1104,7 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)

 static inline struct task *ckpt_init_task(struct ckpt_ctx *ctx)
 {
- return (&ctx->tasks_arr[0]);
+ return (&ctx->tasks[0]);
 }

 /*
```

```
@@ -1123,8 +1123,8 @@ static int ckpt_build_tree(struct ckpt_ctx *ctx)
   * placeholder tasks (each session id may have at most one)
   */
  ctx->tasks_max = ctx->pids_nr * 4;
- ctx->tasks_arr = malloc(sizeof(*ctx->tasks_arr) * ctx->tasks_max);
- if (!ctx->tasks_arr) {
+ ctx->tasks = malloc(sizeof(*ctx->tasks) * ctx->tasks_max);
+ if (!ctx->tasks) {
   ckpt_perror("malloc tasks array");
   return -1;
  }
@@ -1135,7 +1135,7 @@ static int ckpt_build_tree(struct ckpt_ctx *ctx)

  /* assign a creator to each task */
  for (i = 0; i < ctx->tasks_nr; i++) {
-  task = &ctx->tasks_arr[i];
+  task = &ctx->tasks[i];
   if (task->creator)
    continue;
   if (ckpt_set_creator(ctx, task) < 0)
@@ -1145,7 +1145,7 @@ static int ckpt_build_tree(struct ckpt_ctx *ctx)
 #ifdef CHECKPOINT_DEBUG
  ckpt_dbg("====== TASKS\n");
  for (i = 0; i < ctx->tasks_nr; i++) {
-  task = &ctx->tasks_arr[i];
+  task = &ctx->tasks[i];
   ckpt_dbg("\t[%d] pid %d ppid %d sid %d creator %d",
     i, task->pid, task->ppid, task->sid,
     task->creator->pid);
@@ -1180,7 +1180,7 @@ static int ckpt_setup_task(struct ckpt_ctx *ctx, pid_t pid, pid_t ppid)
  if (hash_lookup(ctx, pid))  /* already handled */
   return 0;

- task = &ctx->tasks_arr[ctx->tasks_nr++];
+ task = &ctx->tasks[ctx->tasks_nr++];

  task->flags = TASK_GHOST;

@@ -1305,7 +1305,7 @@ static int ckpt_init_tree(struct ckpt_ctx *ctx)

  /* populate with known tasks */
  for (i = 0; i < pids_nr; i++) {
-  task = &ctx->tasks_arr[i];
+  task = &ctx->tasks[i];

   task->flags = 0;

@@ -1582,7 +1582,7 @@ static int ckpt_set_creator(struct ckpt_ctx *ctx, struct task *task)
```

```
 static int ckpt_placeholder_task(struct ckpt_ctx *ctx, struct task *task)
 {
  struct task *session = hash_lookup(ctx, task->sid);
- struct task *holder = &ctx->tasks_arr[ctx->tasks_nr++];
+ struct task *holder = &ctx->tasks[ctx->tasks_nr++];
  pid_t pid;

  if (ctx->tasks_nr > ctx->tasks_max) {
@@ -2492,7 +2492,7 @@ static int assign_vpids(struct ckpt_ctx *ctx)
  }

  for (tidx = 0, hidx = 0, vidx = 0; tidx < ctx->pids_nr; tidx++) {
-  task = &ctx->tasks_arr[tidx];
+  task = &ctx->tasks[tidx];
   depth = ctx->pids_arr[tidx].depth;

   task->vidx = vidx;
@@ -2532,7 +2532,7 @@ static int ckpt_read_vpids(struct ckpt_ctx *ctx)
   if (ctx->pids_arr[i].depth < 0) {
    ckpt_err("Invalid depth %d for pid %d",
      ctx->pids_arr[i].depth,
-     ctx->tasks_arr[i].pid);
+     ctx->tasks[i].pid);
    errno = -EINVAL;
    return -1;
   }
--
1.7.1
```