If we are doing a self-checkpoint, we will not be frozen and the get_freezer_task() will return NULL. If we have children, then may_checkpoint_task() will attempt to make sure that they are in the same cgroup as the freezer task, which will attempt to lock a NULL task pointer (and thus go boom).

This patch just inserts a check before in_same_cgroup_freezer() call to make sure we have a freezer task. The existing check is aimed to determine if a task is not frozen alongside the root, so this addition catches the case where it is impossible.

Signed-off-by: Dan Smith <danms@us.ibm.com>
---
kernel/checkpoint/checkpoint.c | 8 ++++++-
1 files changed, 6 insertions(+), 2 deletions(-)

diff --git a/kernel/checkpoint/checkpoint.c b/kernel/checkpoint/checkpoint.c
index 0f46acf..33f53e9 100644
--- a/kernel/checkpoint/checkpoint.c
+++ b/kernel/checkpoint/checkpoint.c
@@ -262,8 +262,12 @@ static int may_checkpoint_task(struct ckpt_ctx *ctx, struct task_struct *t)
if (t->exit_state)
return 0;
-/* verify that all tasks belongs to same freezer cgroup */
-if (t != current && !in_same_cgroup_freezer(t, ctx->root_freezer)) {
+/*
+ * verify that we have a freezer cgroup and that all tasks
+ * belong to the same one
+ */
+if (t != current && (!ctx->root_freezer ||
+!in_same_cgroup_freezer(t, ctx->root_freezer))) {
+_ckpt_err(ctx, -EBUSY, "%T)Not frozen or wrong cgroup\n");
return -EBUSY;
}