
Subject: [PATCH] ia64 sn xpc: Convert to use kthread API.

Posted by [ebiederm](#) on Thu, 19 Apr 2007 07:58:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: Eric W. Biederman <ebiederm@xmission.com>

This patch starts the xpc kernel threads using kthread_run not a combination of kernel_thread and daemonize. Resulting in slightly simpler and more maintainable code.

Cc: Jes Sorensen <jes@sgi.com>

Cc: Tony Luck <tony.luck@intel.com>

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

arch/ia64/sn/kernel/xpc_main.c | 31 ++++++++-----
1 files changed, 13 insertions(+), 18 deletions(-)

diff --git a/arch/ia64/sn/kernel/xpc_main.c b/arch/ia64/sn/kernel/xpc_main.c

index e336e16..5b53642 100644

--- a/arch/ia64/sn/kernel/xpc_main.c

+++ b/arch/ia64/sn/kernel/xpc_main.c

@@ -56,6 +56,7 @@

#include <linux/reboot.h>

#include <linux/completion.h>

#include <linux/kdebug.h>

+#include <linux/kthread.h>

#include <asm/sn/intr.h>

#include <asm/sn/sn_sal.h>

#include <asm/uaccess.h>

@@ -253,8 +254,6 @@ xpc_hb_checker(void *ignore)

/* this thread was marked active by xpc_hb_init() */

- daemonize(XPC_HB_CHECK_THREAD_NAME);

-

set_cpus_allowed(current, cpumask_of_cpu(XPC_HB_CHECK_CPU));

xpc_hb_check_timeout = jiffies + (xpc_hb_check_interval * HZ);

@@ -324,8 +323,6 @@ xpc_hb_checker(void *ignore)

static int

xpc_initiate_discovery(void *ignore)

{

- daemonize(XPC_DISCOVERY_THREAD_NAME);

-

xpc_discovery();

dev_dbg(xpc_part, "discovery thread is exiting\n");

@@ -494,8 +491,6 @@ xpc_activating(void *__partid)

```

dev_dbg(xpc_part, "bringing partition %d up\n", partid);

- daemonize("xpc%02d", partid);
-
/*
 * This thread needs to run at a realtime priority to prevent a
 * significant performance degradation.
@@ -559,7 +554,7 @@ xpc_activate_partition(struct xpc_partition *part)
{
    partid_t partid = XPC_PARTID(part);
    unsigned long irq_flags;
-    pid_t pid;
+    struct task_struct *task;

    spin_lock_irqsave(&part->act_lock, irq_flags);
@@ -571,9 +566,10 @@ xpc_activate_partition(struct xpc_partition *part)

    spin_unlock_irqrestore(&part->act_lock, irq_flags);

-    pid = kernel_thread(xpc_activating, (void *) ((u64) partid), 0);
+    task = kthread_run(xpc_activating, (void *) ((u64) partid),
+        "xpc%02d", partid);

-    if (unlikely(pid <= 0)) {
+    if (unlikely(IS_ERR(task))) {
        spin_lock_irqsave(&part->act_lock, irq_flags);
        part->act_state = XPC_P_INACTIVE;
        XPC_SET_REASON(part, xpcCloneKThreadFailed, __LINE__);
@@ -724,8 +720,6 @@ xpc_daemonize_kthread(void *args)
    unsigned long irq_flags;

-    daemonize("xpc%02dc%d", partid, ch_number);
-
    dev_dbg(xpc_chan, "kthread starting, partid=%d, channel=%d\n",
        partid, ch_number);

@@ -844,8 +838,9 @@ xpc_create_kthreads(struct xpc_channel *ch, int needed,
    (void) xpc_part_ref(part);
    xpc_msgqueue_ref(ch);

-    pid = kernel_thread(xpc_daemonize_kthread, (void *) args, 0);
-    if (pid < 0) {
+    task = kthread_run(xpc_daemonize_kthread, args,
+        "xpc%02dc%d", partid, ch_number);
+    if (IS_ERR(task)) {

```

```

/* the fork failed */

/*
@@ -1222,7 +1217,7 @@ xpc_init(void)
    int ret;
    partid_t partid;
    struct xpc_partition *part;
-   pid_t pid;
+   struct task_struct *task;
    size_t buf_size;

@@ -1353,8 +1348,8 @@ xpc_init(void)
    * The real work-horse behind xpc. This processes incoming
    * interrupts and monitors remote heartbeats.
    */
-   pid = kernel_thread(xpc_hb_checker, NULL, 0);
-   if (pid < 0) {
+   task = kthread_run(xpc_hb_checker, NULL, XPC_HB_CHECK_THREAD_NAME);
+   if (IS_ERR(task)) {
        dev_err(xpc_part, "failed while forking hb check thread\n");

        /* indicate to others that our reserved page is uninitialized */
@@ -1384,8 +1379,8 @@ xpc_init(void)
    * activate based on info provided by SAL. This new thread is short
    * lived and will exit once discovery is complete.
    */
-   pid = kernel_thread(xpc_initiate_discovery, NULL, 0);
-   if (pid < 0) {
+   task = kthread_run(xpc_initiate_discovery, NULL, XPC_DISCOVERY_THREAD_NAME);
+   if (IS_ERR(task)) {
        dev_err(xpc_part, "failed while forking discovery thread\n");

        /* mark this new thread as a non-starter */
--
1.5.0.g53756

```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
