
Subject: Re: [PATCH] ia64 sn xpc: Convert to use kthread API.
Posted by [Christoph Hellwig](#) on Sun, 22 Apr 2007 20:36:47 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Thu, Apr 19, 2007 at 01:58:44AM -0600, Eric W. Biederman wrote:

> 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.

This driver is a really twisted maze. It has a lot of threads,
some of them running through the whole lifetime of the driver,
some short-lived and some in a sort of a pool.

The patch below fixes up the long-lived thread as well as fixing
gazillions of leaks in the init routine by switching to proper
goto-based unwinding.

Note that thread pools are something we have in a few places,
and might be worth handling in the core kthread infrastructure,
as tearing down pools will get a bit complicated using the
kthread APIs.

Signed-off-by: Christoph Hellwig <hch@lst.de>

Index: linux-2.6/arch/ia64/sn/kernel/xpc_main.c

```
=====
--- linux-2.6.orig/arch/ia64/sn/kernel/xpc_main.c 2007-04-22 21:19:22.000000000 +0200
+++ linux-2.6/arch/ia64/sn/kernel/xpc_main.c 2007-04-22 21:33:54.000000000 +0200
@@ -55,6 +55,7 @@
#include <linux/delay.h>
#include <linux/reboot.h>
#include <linux/completion.h>
+#include <linux/kthread.h>
#include <asm/sn/intr.h>
#include <asm/sn/sn_sal.h>
#include <asm/kdebug.h>
@@ -159,16 +160,14 @@ static struct ctl_table_header *xpc_sysc
int xpc_disengage_request_timedout;

/* #of IRQs received */
-static atomic_t xpc_act_IRQ_rcvd;
+static atomic_t xpc_act_IRQ_rcvd = ATOMIC_INIT(0);

/* IRQ handler notifies this wait queue on receipt of an IRQ */
static DECLARE_WAIT_QUEUE_HEAD(xpc_act_IRQ_wq);
```

```

+static struct task_struct *xpc_hb_checker_thread;
static unsigned long xpc_hb_check_timeout;

-/* notification that the xpc_hb_checker thread has exited */
-static DECLARE_COMPLETION(xpc_hb_checker_exited);
-
/* notification that the xpc_discovery thread has exited */
static DECLARE_COMPLETION(xpc_discovery_exited);

@@ -250,17 +249,10 @@ xpc_hb_checker(void *ignore)
    int new_IRQ_count;
    int force_IRQ=0;

-
    /* 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);

- while (!(volatile int) xpc_exiting) {
-
+ while (!kthread_should_stop()) {
    dev_dbg(xpc_part, "woke up with %d ticks rem; %d IRQs have "
        "been received\n",
        (int) (xpc_hb_check_timeout - jiffies),
@@ -304,14 +296,10 @@ xpc_hb_checker(void *ignore)
    (void) wait_event_interruptible(xpc_act_IRQ_wq,
        (last_IRQ_count < atomic_read(&xpc_act_IRQ_rcvd) ||
        jiffies >= xpc_hb_check_timeout ||
-        (volatile int) xpc_exiting));
+        kthread_should_stop()));
    }

    dev_dbg(xpc_part, "heartbeat checker is exiting\n");
-
-
- /* mark this thread as having exited */
- complete(&xpc_hb_checker_exited);
    return 0;
}

@@ -966,9 +954,7 @@ xpc_do_exit(enum xpc_retval reason)
    /* wait for the discovery thread to exit */
    wait_for_completion(&xpc_discovery_exited);

```

```

- /* wait for the heartbeat checker thread to exit */
- wait_for_completion(&xpc_hb_checker_exited);
-
+ kthread_stop(xpc_hb_checker_thread);

/* sleep for a 1/3 of a second or so */
(void) msleep_interruptible(300);
@@ -1219,29 +1205,29 @@ xpc_system_die(struct notifier_block *nb
int __init
xpc_init(void)
{
- int ret;
+ int ret = -ENODEV;
    partid_t partid;
    struct xpc_partition *part;
    pid_t pid;
    size_t buf_size;

+ if (!lia64_platform_is("sn2"))
+ goto out;

- if (!lia64_platform_is("sn2")) {
- return -ENODEV;
- }
-
+ ret = -ENOMEM;
    buf_size = max(XPC_RP_VARS_SIZE,
        XPC_RP_HEADER_SIZE + XP_NASID_MASK_BYTES);
    xpc_remote_copy_buffer = xpc_kmalloc_cacheline_aligned(buf_size,
        GFP_KERNEL, &xpc_remote_copy_buffer_base);
- if (xpc_remote_copy_buffer == NULL)
- return -ENOMEM;
+ if (!xpc_remote_copy_buffer)
+ goto out;

    snprintf(xpc_part->bus_id, BUS_ID_SIZE, "part");
    snprintf(xpc_chan->bus_id, BUS_ID_SIZE, "chan");

    xpc_sysctl = register_sysctl_table(xpc_sys_dir);
+ if (!xpc_sysctl)
+ goto out_free_remote_buffer;

/*
 * The first few fields of each entry of xpc_partitions[] need to
@@ -1278,12 +1264,6 @@ xpc_init(void)
    xpc_allow_IPI_ops();

```

```

/*
- * Interrupts being processed will increment this atomic variable and
- * awaken the heartbeat thread which will process the interrupts.
- */
- atomic_set(&xpc_act_IRQ_rcvd, 0);
-
- /*
-  * This is safe to do before the xpc_hb_checker thread has started
-  * because the handler releases a wait queue. If an interrupt is
-  * received before the thread is waiting, it will not go to sleep,
@@ -1294,15 +1274,7 @@ xpc_init(void)
if (ret != 0) {
dev_err(xpc_part, "can't register ACTIVATE IRQ handler, "
"errno=%d\n", -ret);
-
- xpc_restrict_IPI_ops();
-
- if (xpc_sysctl) {
- unregister_sysctl_table(xpc_sysctl);
- }
-
- kfree(xpc_remote_copy_buffer_base);
- return -EBUSY;
+ goto out_restrict_IPI_ops;
}

/*
@@ -1313,29 +1285,23 @@ xpc_init(void)
xpc_rsvd_page = xpc_rsvd_page_init();
if (xpc_rsvd_page == NULL) {
dev_err(xpc_part, "could not setup our reserved page\n");
-
- free_irq(SGI_XPC_ACTIVATE, NULL);
- xpc_restrict_IPI_ops();
-
- if (xpc_sysctl) {
- unregister_sysctl_table(xpc_sysctl);
- }
-
- kfree(xpc_remote_copy_buffer_base);
- return -EBUSY;
+ ret = -ENOMEM;
+ goto out_free_irq;
}

/* add ourselves to the reboot_notifier_list */

```

```

    ret = register_reboot_notifier(&xpc_reboot_notifier);
    if (ret != 0) {
-   dev_warn(xpc_part, "can't register reboot notifier\n");
+   dev_err(xpc_part, "can't register reboot notifier\n");
+   goto out_free_rsvd_page;
    }

    /* add ourselves to the die_notifier list (i.e., ia64die_chain) */
    ret = register_die_notifier(&xpc_die_notifier);
    if (ret != 0) {
-   dev_warn(xpc_part, "can't register die notifier\n");
+   dev_err(xpc_part, "can't register die notifier\n");
+   goto out_unregister_reboot_notifier;
    }

@@ -1353,31 +1319,16 @@ 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) {
+   xpc_hb_checker_thread = kthread_create(xpc_hb_checker, NULL,
+   XPC_HB_CHECK_THREAD_NAME);
+   if (IS_ERR(xpc_hb_checker_thread)) {
        dev_err(xpc_part, "failed while forking hb check thread\n");
-   }
-   /* indicate to others that our reserved page is uninitialized */
-   xpc_rsvd_page->vars_pa = 0;
-   /* take ourselves off of the reboot_notifier_list */
-   (void) unregister_reboot_notifier(&xpc_reboot_notifier);
-   /* take ourselves off of the die_notifier list */
-   (void) unregister_die_notifier(&xpc_die_notifier);
-   del_timer_sync(&xpc_hb_timer);
-   free_irq(SGI_XPC_ACTIVATE, NULL);
-   xpc_restrict_IPI_ops();
-   if (xpc_sysctl) {
-   unregister_sysctl_table(xpc_sysctl);
-   }
-   kfree(xpc_remote_copy_buffer_base);
-   return -EBUSY;
+   ret = PTR_ERR(xpc_hb_checker_thread);
+   goto out_del_hb_timer;

```

```

}

+ kthread_bind(xpc_hb_checker_thread, XPC_HB_CHECK_CPU);
+ wake_up_process(xpc_hb_checker_thread);

/*
 * Startup a thread that will attempt to discover other partitions to
@@ -1403,6 +1354,29 @@ xpc_init(void)
    xpc_initiate_partid_to_nasids);

    return 0;
+
+ if (ret != 0) {
+   dev_err(xpc_part, "can't register reboot notifier\n");
+   goto out_free_rsvd_page;
+ }
+
+ out_del_hb_timer:
+ unregister_die_notifier(&xpc_die_notifier);
+ out_unregister_reboot_notifier:
+ unregister_reboot_notifier(&xpc_reboot_notifier);
+ out_free_rsvd_page:
+ /* indicate to others that our reserved page is uninitialized */
+ xpc_rsvd_page->vars_pa = 0;
+ /* XXX(hch): xpc_rsvd_page gets leaked */
+ out_free_irq:
+ free_irq(SGI_XPC_ACTIVATE, NULL);
+ out_restrict_IPI_ops:
+ xpc_restrict_IPI_ops();
+ unregister_sysctl_table(xpc_sysctl);
+ out_free_remote_buffer:
+ kfree(xpc_remote_copy_buffer_base);
+ out:
+ return -EBUSY;
}
module_init(xpc_init);

```

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