

[PATCH 2/2] Module removal to use kthread

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-01/7612.html>

From: Rusty Russell (rusty_at_rustcorp.com.au)

Date: 01/30/04

To: akpm@osdl.org

Date: Fri, 30 Jan 2004 16:51:45 +1100

[-mm tree only, since it needs kthread]

module.c currently spawns threads directly to stop the machine, so a module can be atomically tested for removal.

Unfortunately, this means that the current task is manipulated (which races with `set_cpus_allowed`, for example), and it can't set its priority artificially high. Using a kernel thread can solve this cleanly, and with `kthread_run`, it's simple.

Name: Use Kthread in module.c

Author: Rusty Russell

Status: Tested on 2.6.2-rc2-mm1

Depends: Hotcpu/kthread-simple.patch.gz

Depends: Hotcpu/cpucontrol_macros.patch.gz

D: module.c currently spawns threads directly to stop the machine, so

D: a module can be atomically tested for removal.

D:

D: Unfortunately, this means that the current task is manipulated

D: (which races with `set_cpus_allowed`, for example), and it can't set

D: its priority artificially high. Using a kernel thread can solve

D: this cleanly, and with `kthread_run`, it's simple.

```
diff -urpN --exclude TAGS -X /home/rusty/devel/kernel/kernel-patches/current-dontdiff --minimal
.4985-linux-2.6.1-bk5/include/linux/sched.h .4985-linux-2.6.1-bk5.updated/include/linux/sched.h
--- .4985-linux-2.6.1-bk5/include/linux/sched.h 2004-01-10 13:59:38.000000000 +1100
+++ .4985-linux-2.6.1-bk5.updated/include/linux/sched.h 2004-01-20 19:27:27.000000000 +1100
@@ -706,6 +706,8 @@ extern task_t *child_reaper;
extern int do_execve(char *, char __user * __user *, char __user * __user *, struct pt_regs *);
extern long do_fork(unsigned long, unsigned long, struct pt_regs *, unsigned long, int __user *, int __user *);
extern struct task_struct * copy_process(unsigned long, unsigned long, struct pt_regs *, unsigned long, int
__user *, int __user *);
+extern asmlinkage long sys_sched_setscheduler(pid_t pid, int policy,
+ struct sched_param __user *parm);
```

Linux-Kernel: [PATCH 2/2] Module removal to use kthread

```
#ifdef CONFIG_SMP
extern void wait_task_inactive(task_t * p);
diff -urpN --exclude TAGS -X /home/rusty/devel/kernel/kernel-patches/current-dontdiff --minimal
.4985-linux-2.6.1-bk5/kernel/module.c .4985-linux-2.6.1-bk5.updated/kernel/module.c
---- .4985-linux-2.6.1-bk5/kernel/module.c 2004-01-10 13:59:39.000000000 +1100
+++ .4985-linux-2.6.1-bk5.updated/kernel/module.c 2004-01-20 19:49:02.000000000 +1100
@@ -32,6 +32,7 @@
#include <linux/err.h>
#include <linux/vermagic.h>
#include <linux/notifier.h>
+#include <linux/kthread.h>
#include <asm/uaccess.h>
#include <asm/semaphore.h>
#include <asm/pgalloc.h>
@@ -457,6 +458,40 @@ static void module_unload_free(struct mo
    }
}

+#ifdef CONFIG_MODULE_FORCE_UNLOAD
+static inline int try_force(unsigned int flags)
+{
+ int ret = (flags & O_TRUNC);
+ if (ret)
+ tainted |= TAINT_FORCED_MODULE;
+ return ret;
+}
+#else
+static inline int try_force(unsigned int flags)
+{
+ return 0;
+}
+#endif /* CONFIG_MODULE_FORCE_UNLOAD */
+
+static int try_stop_module_local(struct module *mod, int flags, int *forced)
+{
+ local_irq_disable();
+
+ /* If it's not unused, quit unless we are told to block. */
+ if ((flags & O_NONBLOCK) && module_refcount(mod) != 0) {
+ if (!(*forced = try_force(flags))) {
+ local_irq_enable();
+ return -EWOULDBLOCK;
+ }
+ }
+
+ /* Mark it as dying. */
+ mod->waiter = current;
+ mod->state = MODULE_STATE_GOING;
+ local_irq_enable();
+ return 0;
+}

```

```

+
+ #ifdef CONFIG_SMP
+ /* Thread to stop each CPU in user context. */
+ enum stopref_state {
@@ -475,13 +510,6 @@ static int stopref(void *cpu)
+     int irqs_disabled = 0;
+     int prepared = 0;

- sprintf(current->comm, "kmodule%lu\n", (unsigned long)cpu);
-
- /* Highest priority we can manage, and move to right CPU. */
- #if 0 /* FIXME */
- struct sched_param param = { .sched_priority = MAX_RT_PRIO-1 };
- setscheduler(current->pid, SCHED_FIFO, &param);
- #endif
+     set_cpus_allowed(current, cpumask_of_cpu((int)(long)cpu));

+     /* Ack: we are alive */
@@ -535,29 +563,30 @@ static void stopref_set_state(enum stopr
+     }
+ }

- /* Stop the machine. Disables irqs. */
- static int stop_refcounts(void)
+ struct stopref
+ {
+     unsigned int i, cpu;
+     cpumask_t old_allowed;
+     int ret = 0;
+     struct module *mod;
+     int flags;
+     int *forced;
+ };

- /* One thread per cpu. We'll do our own. */
- cpu = smp_processor_id();
+ static int spawn_stopref(void *data)
+ {
+     struct stopref *sref = data;
+     struct sched_param param = { .sched_priority = MAX_RT_PRIO-1 };
+     unsigned int i, cpu = smp_processor_id();
+     int ret = 0;

- /* FIXME: racy with set_cpus_allowed. */
- old_allowed = current->cpus_allowed;
+ /* One high-prio thread per cpu. We'll do one (any one). */
+     set_cpus_allowed(current, cpumask_of_cpu(cpu));
+     sys_sched_setscheduler(current->pid, SCHED_FIFO, &param);

+     atomic_set(&stopref_thread_ack, 0);
+     stopref_num_threads = 0;

```

Linux-Kernel: [PATCH 2/2] Module removal to use kthread

```
stopref_state = STOPREF_WAIT;

- /* No CPUs can come up or down during this. */
- lock_cpu_hotplug();
-
- for (i = 0; i < NR_CPUS; i++) {
- if (i == cpu || !cpu_online(i))
+ for_each_online_cpu(i) {
+ if (i == cpu)
        continue;
        ret = kernel_thread(stopref, (void *) (long) i, CLONE_KERNEL);
        if (ret < 0)
@@ -572,40 +601,52 @@ static int stop_refcounts(void)
    /* If some failed, kill them all. */
    if (ret < 0) {
        stopref_set_state(STOPREF_EXIT, 1);
- unlock_cpu_hotplug();
        return ret;
    }

    /* Don't schedule us away at this point, please. */
    preempt_disable();

- /* Now they are all scheduled, make them hold the CPUs, ready. */
+ /* Now they are all started, make them hold the CPUs, ready. */
    stopref_set_state(STOPREF_PREPARE, 0);

    /* Make them disable irqs. */
    stopref_set_state(STOPREF_DISABLE_IRQ, 0);

- local_irq_disable();
- return 0;
- }
+ /* Atomically disable module if possible */
+ ret = try_stop_module_local(sref->mod, sref->flags, sref->forced);

- /* Restart the machine. Re-enables irqs. */
- static void restart_refcounts(void)
- {
    stopref_set_state(STOPREF_EXIT, 0);
- local_irq_enable();
    preempt_enable();
- unlock_cpu_hotplug();
+
+ /* Wait for kthread_stop */
+ while (!signal_pending(current)) {
+ __set_current_state(TASK_INTERRUPTIBLE);
+ schedule();
+ }
+ return ret;
}
}
```

Linux-Kernel: [PATCH 2/2] Module removal to use kthread

```

-#else /* ...!SMP */
-static inline int stop_refcounts(void)
+
+static int try_stop_module(struct module *mod, int flags, int *forced)
+ {
+     - local_irq_disable();
+     - return 0;
+     + struct task_struct *p;
+     + struct stopref sref = { mod, flags, forced };
+     + int ret;
+
+     +
+     + /* No CPUs can come up or down during this. */
+     + lock_cpu_hotplug();
+     + p = kthread_run(spawn_stopref, &sref, "krmmod");
+     + if (IS_ERR(p))
+     +     ret = PTR_ERR(p);
+     + else
+     +     ret = kthread_stop(p);
+     + unlock_cpu_hotplug();
+     + return ret;
+ }
-static inline void restart_refcounts(void)
+#else /* ...!SMP */
+static inline int try_stop_module(struct module *mod, int flags, int *forced)
+ {
+     - local_irq_enable();
+     + return try_stop_module_local(mod, flags, forced);
+ }
+ #endif

@@ -622,21 +663,6 @@ EXPORT_SYMBOL(module_refcount);
/* This exists whether we can unload or not */
static void free_module(struct module *mod);

-#ifndef CONFIG_MODULE_FORCE_UNLOAD
-static inline int try_force(unsigned int flags)
- {
-     - int ret = (flags & O_TRUNC);
-     - if (ret)
-     -     tainted |= TAINT_FORCED_MODULE;
-     - return ret;
- }
-#else
-static inline int try_force(unsigned int flags)
- {
-     - return 0;
- }
-#endif /* CONFIG_MODULE_FORCE_UNLOAD */
-
/* Stub function for modules which don't have an exitfn */
void cleanup_module(void)

```

Linux-Kernel: [PATCH 2/2] Module removal to use kthread

```
{
@@ -706,26 +732,9 @@ sys_delete_module(const char __user *nam
        goto out;
    }
}

- /* Stop the machine so refcounts can't move: irqs disabled. */
- DEBUGP("Stopping refcounts...\n");
- ret = stop_refcounts();
- if (ret != 0)
- goto out;
-
- /* If it's not unused, quit unless we are told to block. */
- if ((flags & O_NONBLOCK) && module_refcount(mod) != 0) {
- forced = try_force(flags);
- if (!forced) {
- ret = -EWOULDBLOCK;
- restart_refcounts();
- goto out;
- }
- }

- /* Mark it as dying. */
- mod->waiter = current;
- mod->state = MODULE_STATE_GOING;
- restart_refcounts();
+ /* Stop the machine so refcounts can't move and disable module. */
+ ret = try_stop_module(mod, flags, &forced);

        /* Never wait if forced. */
        if (!forced && module_refcount(mod) != 0)

--
    Anyone who quotes me in their sig is an idiot. -- Rusty Russell.
-
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org
More majordomo info at http://vger.kernel.org/majordomo-info.html
Please read the FAQ at http://www.tux.org/lkml/
```