

Timtertop v7 for dynticks

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-11/9606.html>

From: Con Kolivas (kernel_at_kolivas.org)

Date: 11/30/05

To: Daniel Petrini <d.pensator@gmail.com>

Date: Wed, 30 Nov 2005 13:15:58 +1100

Hi Daniel et al

I finally had a good look at your timertop patch.

Thanks for your work on timer top. I hope you don't mind I've taken the liberty of using your last timertop6 patch and reworked it somewhat.

I've removed the ifdefs from kernel/timer.c and removed some redundant structs (probably lying around from previous versions).

Your account_timer function seems to return an int but that was not used in any way. I've left that intact in case you had other plans, but it takes less parameters now.

I've moved the use of flags to within each function – you cannot use flags across functions

I've changed the allocation of memory to kmem which allows for tracking of the memory usage in slabinfo. I wonder if you should clear all the entries once you stop recording as well.

Lots of general cleanups.

Here is the updated version of timertop; I took the liberty of incrementing the version number I hope you don't mind.

I should be able to include this in the next rolled up dynticks as well unless you had further changes planned.

Cheers,

Con

```
---
include/linux/dyn-tick.h | 14 ++
kernel/Makefile          | 1
kernel/timer.c           | 3
kernel/timer_top.c       | 237 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
lib/Kconfig.debug        | 13 ++
5 files changed, 268 insertions(+)
```

Linux-Kernel: Timertop v7 for dynticks

```
Index: linux-2.6.15-rc3-ck1/kernel/Makefile
=====
--- linux-2.6.15-rc3-ck1.orig/kernel/Makefile    2005-11-30 09:28:26.000000000 +1100
+++ linux-2.6.15-rc3-ck1/kernel/Makefile        2005-11-30 09:28:27.000000000 +1100
@@ -33,6 +33,7 @@ obj-$(CONFIG_CRASH_DUMP) += crash_dump.o
obj-$(CONFIG_SECCOMP) += seccomp.o
obj-$(CONFIG_RCU_TORTURE_TEST) += rcutorture.o
obj-$(CONFIG_NO_IDLE_HZ) += dyn-tick.o
+obj-$(CONFIG_TIMER_INFO) += timer_top.o

ifneq ($(CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER),y)
# According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
Index: linux-2.6.15-rc3-ck1/kernel/timer.c
=====
--- linux-2.6.15-rc3-ck1.orig/kernel/timer.c    2005-11-30 09:28:26.000000000 +1100
+++ linux-2.6.15-rc3-ck1/kernel/timer.c        2005-11-30 12:59:59.000000000 +1100
@@ -33,6 +33,7 @@
#include <linux/posix-timers.h>
#include <linux/cpu.h>
#include <linux/syscalls.h>
+#include <linux/dyn-tick.h>

#include <asm/uaccess.h>
#include <asm/unistd.h>
@@ -540,6 +541,8 @@ found:
                                expires = nte->expires;
                                }
}
+ account_timer((unsigned long)nte->function, nte->data);
+
spin_unlock(&base->t_base.lock);
return expires;
}
Index: linux-2.6.15-rc3-ck1/kernel/timer_top.c
=====
--- /dev/null    1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.15-rc3-ck1/kernel/timer_top.c    2005-11-30 13:09:43.000000000 +1100
@@ -0,0 +1,237 @@
+/*
+ * kernel/timer_top.c
+ *
+ * Export Timers information to /proc/timer_info
+ *
+ * Copyright (C) 2005 Instituto Nokia de Tecnologia - INdT - Manaus
+ * Written by Daniel Pettrini <d.pensator@gmail.com>
+ *
+ * This utility should be used to get information from the system timers
+ * and maybe optimize the system once you know which timers are being used
+ * and the process which starts them.
+ * This is particular useful above dynamic tick implementation. One can
+ * see who is starting timers and make the HZ value increase.
+ *
+ * We export the addresses and counting of timer functions being called,
+ * the pid and cmdline from the owner process if applicable.
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+#include <linux/list.h>
```

Linux-Kernel: Timertop v7 for dynticks

```
+#include <linux/proc_fs.h>
+#include <linux/module.h>
+#include <linux/spinlock.h>
+#include <linux/sched.h>
+#include <linux/seq_file.h>
+#include <asm/uaccess.h>
+
+#define VERSION          "Timer Top v0.9.7"
+
+struct timer_top_info {
+    unsigned int          func_pointer;
+    unsigned long         counter;
+    pid_t                 pid;
+    char                  comm[TASK_COMM_LEN];
+    struct list_head      list;
+};
+
+struct timer_top_root {
+    spinlock_t            lock;
+    struct list_head      list;
+    kmem_cache_t          *cache;
+};
+
+static struct timer_top_root top_root = {
+    .lock          = SPIN_LOCK_UNLOCKED,
+    .list          = LIST_HEAD_INIT(top_root.list),
+};
+
+static struct list_head *timer_list = &top_root.list;
+
+static spinlock_t *top_lock = &top_root.lock;
+static int start_flag = 0;    /* signs if will collect data or not */
+
+static inline struct timer_top_info *update_top_info(unsigned long function,
+    pid_t pid_info)
+{
+    struct timer_top_info *top;
+
+    list_for_each_entry(top, timer_list, list) {
+        /* if it is in the list increment its count */
+        if (top->func_pointer == function && top->pid == pid_info) {
+            top->counter++;
+            goto out;
+        }
+    }
+
+    top = NULL;
+out:
+    return top;
+}
+
+int account_timer(unsigned long function, unsigned long data)
+{
+    struct timer_top_info *top;
+    struct task_struct * task_info;
+    pid_t pid_info = 0;
+    char name[TASK_COMM_LEN] = "";
+    unsigned long flags;
+
+    if (start_flag == 0)
+        goto out;
+}
```

Linux-Kernel: Timertop v7 for dynticks

```
+ spin_lock_irqsave(top_lock, flags);
+
+ if (data) {
+     task_info = (struct task_struct *) data;
+     /* little sanity ... not enough yet */
+     if ((task_info->pid > 0) && (task_info->pid < PID_MAX_LIMIT)) {
+         pid_info = task_info->pid;
+         strncpy(name, task_info->comm, sizeof(task_info->comm));
+     }
+ }
+
+ top = update_top_info(function, pid_info);
+ if (top)
+     goto out_unlock;
+
+ top = kmem_cache_alloc(top_root.cache, GFP_ATOMIC);
+ if (unlikely(!top))
+     goto out_unlock;
+
+ /* if you are here then it didnt find so inserts in the list */
+ top->func_pointer = function;
+ top->counter = 1;
+ top->pid = pid_info;
+ strncpy(top->comm, name, sizeof(name));
+ list_add(&top->list, timer_list);
+
+out_unlock:
+ spin_unlock_irqrestore(top_lock, flags);
+
+out:
+ return 0;
+}
+
+EXPORT_SYMBOL(account_timer);
+
+int timer_list_del(void)
+{
+    struct list_head *aux1, *aux2;
+    struct timer_top_info *entry;
+    unsigned long flags;
+
+    spin_lock_irqsave(top_lock, flags);
+    list_for_each_safe(aux1, aux2, timer_list) {
+        entry = list_entry(aux1, struct timer_top_info, list);
+        list_del(aux1);
+        kmem_cache_free(top_root.cache, entry);
+    }
+    spin_unlock_irqrestore(top_lock, flags);
+    return 0;
+}
+
+/* PROC_FS_SECTION */
+
+static struct proc_dir_entry *top_info_file;
+static struct proc_dir_entry *top_info_file_out;
+
+/* Statistics output - timer_info*/
+static int proc_read_top_info(struct seq_file *m, void *v)
+{
+    struct timer_top_info *top;
+
+    seq_printf(m, "Function counter - %s\n", VERSION);
+}
```

Linux-Kernel: Timertop v7 for dynticks

```
+
+   list_for_each_entry(top, timer_list, list) {
+       seq_printf(m, "%x %lu %d %s\n", top->func_pointer, top->counter, top->pid, top->name);
+   }
+
+   if (start_flag == 0) {
+       seq_printf(m, "Disabled\n");
+   }
+
+   return 0;
+}
+
+static int proc_timertop_open(struct inode *inode, struct file *file)
+{
+   return single_open(file, proc_read_top_info, NULL);
+}
+
+static struct file_operations proc_timertop_operations = {
+   .open           = proc_timertop_open,
+   .read           = seq_read,
+   .llseek        = seq_lseek,
+   .release        = single_release,
+};
+
+#define MAX_INPUT_TOP 10
+
+/* Receive some commands from user - timer_input */
+static int proc_write_timer_input(struct file *file, const char *page,
+                                  unsigned long count, void *data)
+{
+   int len;
+   char input_data[MAX_INPUT_TOP];
+
+   /* input size checking */
+   if(count > MAX_INPUT_TOP - 1)
+       len = MAX_INPUT_TOP - 1;
+   else
+       len = count;
+
+   if (copy_from_user(input_data, page, len))
+       return -EFAULT;
+
+   input_data[len] = '\0';
+
+   if(!strncmp(input_data, "clear", 5))
+       timer_list_del();
+   else if(!strncmp(input_data, "start", 5))
+       start_flag = 1;
+   else if(!strncmp(input_data, "stop", 4))
+       start_flag = 0;
+
+   return len;
+}
+
+/* Print a sample string showing the possible inputs - timer_input */
+static int proc_read_timer_input(char *page, char **start, off_t off,
+                                  int count, int *eof, void *data)
+{
+   int len;
+
+   len = sprintf(page, "clear start stop\n");
+}
```

Linux-Kernel: Timertop v7 for dynticks

```
+     return len;
+}
+
+static int __init init_top_info(void)
+{
+     top_root.cache = kmem_cache_create("top_info",
+     sizeof(struct timer_top_info), 0, SLAB_PANIC, NULL, NULL);
+
+     top_info_file = create_proc_entry("timer_info", 0444, NULL);
+     if (top_info_file == NULL)
+         return -ENOMEM;
+
+     top_info_file_out = create_proc_entry("timer_input", 0666, NULL);
+     if (top_info_file_out == NULL)
+         return -ENOMEM;
+
+     /* Statistics output */
+     top_info_file->proc_fops = &proc_timertop_operations;
+
+     /* Control */
+     top_info_file_out->write_proc = &proc_write_timer_input;
+     top_info_file_out->read_proc = &proc_read_timer_input;
+
+     return 0;
+}
+
+module_init(init_top_info);
Index: linux-2.6.15-rc3-ck1/lib/Kconfig.debug
=====
--- linux-2.6.15-rc3-ck1.orig/lib/Kconfig.debug 2005-11-30 09:28:19.000000000 +1100
+++ linux-2.6.15-rc3-ck1/lib/Kconfig.debug      2005-11-30 09:28:27.000000000 +1100
@@ -77,6 +77,19 @@ config SCHEDSTATS
     application, you can say N to avoid the very slight overhead
     this adds.

+config TIMER_INFO
+     bool "Collect kernel timers statistics"
+     depends on DEBUG_KERNEL && PROC_FS && NO_IDLE_HZ
+     help
+     If you say Y here, additional code will be inserted into the
+     timer routines to collect statistics about kernel timers being
+     reprogrammed through dynamic ticks feature. This statistics
+     will be provided in /proc/timer_info and the behavior of this
+     feature can be controlled through /proc/timer_input.
+     The goal is to offer some output to let user applications show
+     timers pattern usage and allow some tuning in them to
+     maximize idle time.
+
     config DEBUG_SLAB
         bool "Debug memory allocations"
         depends on DEBUG_KERNEL
Index: linux-2.6.15-rc3-ck1/include/linux/dyn-tick.h
=====
--- linux-2.6.15-rc3-ck1.orig/include/linux/dyn-tick.h 2005-11-30 12:59:34.000000000 +1100
+++ linux-2.6.15-rc3-ck1/include/linux/dyn-tick.h      2005-11-30 12:59:59.000000000 +1100
@@ -51,6 +51,15 @@ extern int dyn_tick_enabled(void);
     extern void timer_dyn_reprogram(void);
     extern void set_dyn_tick_max_skip(unsigned int max_skip);

+#ifdef CONFIG_TIMER_INFO
+extern int account_timer(unsigned long function, unsigned long data);
+#else
```

Linux-Kernel: Timertop v7 for dynticks

```
+static inline int account_timer(unsigned long function, unsigned long data)
+{
+    return 0;
+}
+#endif
+
+/* CONFIG_NO_IDLE_HZ */
+static inline int dyn_tick_enabled(void)
+{
@@ -60,6 +69,11 @@ static inline int dyn_tick_enabled(void)
+static inline void set_dyn_tick_max_skip(unsigned int max_skip)
+{
+}
+
+static inline int account_timer(unsigned long function, unsigned long data)
+{
+    return 0;
+}
+#endif /* CONFIG_NO_IDLE_HZ */
+
+/* Pick up arch specific header */
-
```

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/>