

---

Subject: [RFC][PATCH] Make (hr)timers use struct pid instead of pid\_t  
Posted by [Pavel Emelianov](#) on Fri, 24 Aug 2007 09:54:10 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

(Just RFC, untested :( )

Timers may store the task's (that armed the timer) pid for statistics (CONFIG\_TIMER\_STATS). Since these ids are shown via proc, we cannot just store the id and use it - we must get the struct pid and print the id, depending on the namespace, reading the file.

The thing I'm afraid of is the pid reference counting. I've looked through the code and found that the stats start/clear calls looks to be called symmetrically, but I'm not 100% sure.

The other problem is the compilation - as you can see the put\_pid() is declared explicitly in timer.h and hrtimer.h. This is done so because the following dependency arises when including the pid.h directly into these headers:

In file included from include/linux/timer.h:93,  
from include/linux/workqueue.h:8,  
from include/linux/slub\_def.h:11,  
from include/linux/slab.h:118,  
from include/linux/percpu.h:5,  
from include/linux/rcupdate.h:41,  
from include/linux/dcache.h:10,  
from include/linux/fs.h:282,  
from init/do\_mounts\_initrd.c:3:

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

---

```
diff --git a/include/linux/hrtimer.h b/include/linux/hrtimer.h
index 540799b..bb991fa 100644
--- a/include/linux/hrtimer.h
+++ b/include/linux/hrtimer.h
@@ -122,7 +122,7 @@ struct hrtimer {
#ifdef CONFIG_TIMER_STATS
    void    *start_site;
    char    start_comm[16];
-   int     start_pid;
+   struct pid *start_pid;
#endif
};

@@ -328,7 +328,7 @@ extern void sysrq_timer_list_show(void);
```

```

*/
#ifdef CONFIG_TIMER_STATS

-extern void timer_stats_update_stats(void *timer, pid_t pid, void *startf,
+extern void timer_stats_update_stats(void *timer, struct pid *pid, void *startf,
    void *timerf, char *comm,
    unsigned int timer_flag);

@@ -346,8 +346,10 @@ static inline void timer_stats_hrtimer_s
__timer_stats_hrtimer_set_start_info(timer, __builtin_return_address(0));
}

+extern void FASTCALL(put_pid(struct pid *pid));
static inline void timer_stats_hrtimer_clear_start_info(struct hrtimer *timer)
{
+ put_pid(timer->start_pid);
timer->start_site = NULL;
}
#else
diff --git a/include/linux/timer.h b/include/linux/timer.h
index 78cf899..6ad58ff 100644
--- a/include/linux/timer.h
+++ b/include/linux/timer.h
@@ -18,7 +18,7 @@ struct timer_list {
#ifdef CONFIG_TIMER_STATS
void *start_site;
char start_comm[16];
- int start_pid;
+ struct pid *start_pid;
#endif
};

@@ -94,7 +94,7 @@ extern unsigned long get_next_timer_inte

extern void init_timer_stats(void);

-extern void timer_stats_update_stats(void *timer, pid_t pid, void *startf,
+extern void timer_stats_update_stats(void *timer, struct pid *pid, void *startf,
    void *timerf, char *comm,
    unsigned int timer_flag);

@@ -106,8 +106,10 @@ static inline void timer_stats_timer_set
__timer_stats_timer_set_start_info(timer, __builtin_return_address(0));
}

+extern void FASTCALL(put_pid(struct pid *pid));
static inline void timer_stats_timer_clear_start_info(struct timer_list *timer)
{

```

```

+ put_pid(timer->start_pid);
  timer->start_site = NULL;
}
#else
diff --git a/kernel/hrtimer.c b/kernel/hrtimer.c
index c21ca6b..603ae6c 100644
--- a/kernel/hrtimer.c
+++ b/kernel/hrtimer.c
@@ -609,7 +609,7 @@ void __timer_stats_hrtimer_set_start_inf

  timer->start_site = addr;
  memcpy(timer->start_comm, current->comm, TASK_COMM_LEN);
- timer->start_pid = current->pid;
+ timer->start_pid = get_pid(task_pid(current));
}
#endif

@@ -973,7 +973,7 @@ void hrtimer_init(struct hrtimer *timer,

#ifdef CONFIG_TIMER_STATS
  timer->start_site = NULL;
- timer->start_pid = -1;
+ timer->start_pid = NULL;
  memset(timer->start_comm, 0, TASK_COMM_LEN);
#endif
}
diff --git a/kernel/time/timer_list.c b/kernel/time/timer_list.c
index fdb2e03..ed05f0d 100644
--- a/kernel/time/timer_list.c
+++ b/kernel/time/timer_list.c
@@ -17,6 +17,7 @@
#include <linux/seq_file.h>
#include <linux/kallsyms.h>
#include <linux/tick.h>
+#include <linux/nsproxy.h>

#include <asm/uaccess.h>

@@ -62,7 +63,8 @@ print_timer(struct seq_file *m, struct h
  print_name_offset(m, timer->start_site);
  memcpy(tmp, timer->start_comm, TASK_COMM_LEN);
  tmp[TASK_COMM_LEN] = 0;
- SEQ_printf(m, " %s/%d", tmp, timer->start_pid);
+ SEQ_printf(m, " %s/%d", tmp,
+ pid_nr_ns(timer->start_pid, current->nsproxy->pid_ns));
#endif
  SEQ_printf(m, "\n");
  SEQ_printf(m, " # expires at %Lu nsecs [in %Lu nsecs]\n",

```

```

diff --git a/kernel/time/timer_stats.c b/kernel/time/timer_stats.c
index 3c38fb5..10efd40 100644
--- a/kernel/time/timer_stats.c
+++ b/kernel/time/timer_stats.c
@@ -42,6 +42,8 @@
#include <linux/sched.h>
#include <linux/seq_file.h>
#include <linux/kallsyms.h>
+#include <linux/nsproxy.h>
+#include <linux/pid.h>

#include <asm/uaccess.h>

@@ -62,7 +64,7 @@ struct entry {
    void *timer;
    void *start_func;
    void *expire_func;
- pid_t pid;
+ struct pid *pid;

    /*
     * Number of timeout events:
@@ -137,7 +139,11 @@ static struct entry *tstat_hash_table[TS

static void reset_entries(void)
{
- nr_entries = 0;
+ while (nr_entries > 0) {
+ nr_entries--;
+ put_pid(entries[nr_entries].pid);
+ }
+
    memset(entries, 0, sizeof(entries));
    memset(tstat_hash_table, 0, sizeof(tstat_hash_table));
    atomic_set(&overflow_count, 0);
@@ -206,6 +212,7 @@ static struct entry *tstat_lookup(struct
    curr->count = 0;
    curr->next = NULL;
    memcpy(curr->comm, comm, TASK_COMM_LEN);
+ get_pid(curr->pid);

    smp_mb(); /* Ensure that curr is initialized before insert */

@@ -231,9 +238,8 @@ static struct entry *tstat_lookup(struct
    * When the timer is already registered, then the event counter is
    * incremented. Otherwise the timer is registered in a free slot.
    */
-void timer_stats_update_stats(void *timer, pid_t pid, void *startf,

```

```

- void *timerf, char *comm,
- unsigned int timer_flag)
+void timer_stats_update_stats(void *timer, struct pid *pid, void *startf,
+ void *timerf, char *comm, unsigned int timer_flag)
{
/*
 * It doesnt matter which lock we take:
@@ -285,6 +291,7 @@ static int tstats_show(struct seq_file *
long events = 0;
ktime_t time;
int i;
+ struct pid_namespace *ns;

mutex_lock(&show_mutex);
/*
@@ -304,15 +311,13 @@ static int tstats_show(struct seq_file *
seq_printf(m, "Overflow: %d entries\n",
atomic_read(&overflow_count));

+ ns = current->nsproxy->pid_ns;
for (i = 0; i < nr_entries; i++) {
entry = entries + i;
- if (entry->timer_flag & TIMER_STATS_FLAG_DEFERRABLE) {
- seq_printf(m, "%4luD, %5d %-16s ",
- entry->count, entry->pid, entry->comm);
- } else {
- seq_printf(m, " %4lu, %5d %-16s ",
- entry->count, entry->pid, entry->comm);
- }
+ seq_printf(m, "%4lu%s, %5d, %-16s ", entry->count,
+ entry->timer_flag &
+ TIMER_STATS_FLAG_DEFERRABLE ? "D" : "",
+ pid_nr_ns(entry->pid, ns), entry->comm);

print_name_offset(m, (unsigned long)entry->start_func);
seq_puts(m, " (");
diff --git a/kernel/timer.c b/kernel/timer.c
index 1d34be0..3459bca 100644
--- a/kernel/timer.c
+++ b/kernel/timer.c
@@ -305,7 +305,7 @@ void __timer_stats_timer_set_start_info(

timer->start_site = addr;
memcpy(timer->start_comm, current->comm, TASK_COMM_LEN);
- timer->start_pid = current->pid;
+ timer->start_pid = get_pid(task_pid(current));
}

```

```
static void timer_stats_account_timer(struct timer_list *timer)
@@ -336,7 +336,7 @@ void fastcall init_timer(struct timer_li
    timer->base = __raw_get_cpu_var(tvec_bases);
#ifdef CONFIG_TIMER_STATS
    timer->start_site = NULL;
- timer->start_pid = -1;
+ timer->start_pid = NULL;
    memset(timer->start_comm, 0, TASK_COMM_LEN);
#endif
}
```

---

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---