Subject: [PATCH 1/7] introduce atomic_dec_and_lock_irqsave()
Posted by dev on Tue, 29 Aug 2006 14:47:35 GMT
View Forum Message <> Reply to Message

Oleg Nesterov noticed to me that the construction like
(used in beancounter patches and free_uid()):

```
  local_irq_save(flags);
  if (atomic_dec_and_lock(&refcnt, &lock))
   ...
```

is not that good for preemtible kernels, since with preemption
spin_lock() can schedule() to reduce latency. However, it won't schedule
if interrupts are disabled.

So this patch introduces atomic_dec_and_lock_irqsave() as a logical
counterpart to atomic_dec_and_lock().

Signed-Off-By: Pavel Emelianov <xemul@sw.ru>
Signed-Off-By: Kirill Korotaev <dev@sw.ru>

```
 include/linux/spinlock.h |    6 ++++++
 kernel/user.c            |    5 +----
 lib/dec_and_lock.c       |   19 +++++++++++++++++++
 3 files changed, 26 insertions(+), 4 deletions(-)

--- ./include/linux/spinlock.h.dlirq 2006-08-28 10:17:35.000000000 +0400
+++ ./include/linux/spinlock.h 2006-08-28 11:22:37.000000000 +0400
@@ -266,6 +266,12 @@ extern int _atomic_dec_and_lock(atomic_t
 #define atomic_dec_and_lock(atomic, lock) \
   __cond_lock(lock, _atomic_dec_and_lock(atomic, lock))

+extern int _atomic_dec_and_lock_irqsave(atomic_t *atomic, spinlock_t *lock,
+  unsigned long *flagsp);
+#define atomic_dec_and_lock_irqsave(atomic, lock, flags) \
+  __cond_lock(lock, \
+   _atomic_dec_and_lock_irqsave(atomic, lock, &flags))
+
 /**
  * spin_can_lock - would spin_trylock() succeed?
  * @lock: the spinlock in question.
--- ./kernel/user.c.dlirq 2006-07-10 12:39:20.000000000 +0400
+++ ./kernel/user.c 2006-08-28 11:08:56.000000000 +0400
@@ -108,15 +108,12 @@ void free_uid(struct user_struct *up)
  if (!up)
   return;
```

```
- local_irq_save(flags);
- if (atomic_dec_and_lock(&up->__count, &uidhash_lock)) {
+ if (atomic_dec_and_lock_irqsave(&up->__count, &uidhash_lock, flags)) {
   uid_hash_remove(up);
   spin_unlock_irqrestore(&uidhash_lock, flags);
   key_put(up->uid_keyring);
   key_put(up->session_keyring);
   kmem_cache_free(uid_cachep, up);
- } else {
-  local_irq_restore(flags);
  }
 }

--- ./lib/dec_and_lock.c.dlirq 2006-04-21 11:59:36.000000000 +0400
+++ ./lib/dec_and_lock.c 2006-08-28 11:22:08.000000000 +0400
@@ -33,3 +33,22 @@ int _atomic_dec_and_lock(atomic_t *atomi
 }

 EXPORT_SYMBOL(_atomic_dec_and_lock);
+
+/*
+ * the same, but takes the lock with _irqsave
+ */
+int _atomic_dec_and_lock_irqsave(atomic_t *atomic, spinlock_t *lock,
+  unsigned long *flagsp)
+{
+#ifdef CONFIG_SMP
+ if (atomic_add_unless(atomic, -1, 1))
+  return 0;
+#endif
+ spin_lock_irqsave(lock, *flagsp);
+ if (atomic_dec_and_test(atomic))
+  return 1;
+ spin_unlock_irqrestore(lock, *flagsp);
+ return 0;
+}
+
+EXPORT_SYMBOL(_atomic_dec_and_lock_irqsave);
```