Subject: [PATCH 10/28] [PREP 10/14] Prepare some pid.c routines for the namespaces support

Posted by Pavel Emelianov on Fri, 15 Jun 2007 16:08:23 GMT

View Forum Message <> Reply to Message

The alloc_pid(), free_pid() and put_pid() are splitted into two parts
 - the generic pid manipulation, like refcount and task hlist heads;
 - manipulations with pids numerical values.

E.g. alloc pid allocates the struct pid and initializes its refcount etc and
calls the alloc_pid_nrs() to allocate the pid numerical ids and hash it into
the hash-tables.

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

```
 pid.c |   90 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++----------------------------------
 1 files changed, 56 insertions(+), 34 deletions(-)

--- ./kernel/pid.c.pidsplit 2007-06-14 15:57:41.000000000 +0400
+++ ./kernel/pid.c 2007-06-14 16:34:28.000000000 +0400
@@ -174,13 +174,64 @@ static int next_pidmap(struct pid_namesp
  return -1;
 }

+#ifndef CONFIG_PID_NS
+static inline int alloc_pid_nrs(struct pid *pid)
+{
+ int nr;
+
+ nr = alloc_pidmap(&init_pid_ns);
+ if (nr < 0)
+  return nr;
+
+ pid->nr = nr;
+ spin_lock_irq(&pidmap_lock);
+ hlist_add_head_rcu(&pid->pid_chain, &pid_hash[pid_hashfn(nr)]);
+ spin_unlock_irq(&pidmap_lock);
+ return 0;
+}
+
+static inline void unhash_pid_nrs(struct pid *pid)
+{
+ /* We can be called with write_lock_irq(&tasklist_lock) held */
+ unsigned long flags;
+
+ spin_lock_irqsave(&pidmap_lock, flags);
```

```
+ hlist_del_rcu(&pid->pid_chain);
+ spin_unlock_irqrestore(&pidmap_lock, flags);
+
+ free_pidmap(&init_pid_ns, pid->nr);
+}
+
+static inline void free_pid_nrs(struct pid *pid)
+{
+}
+
+struct pid * fastcall find_pid_ns(int nr, struct pid_namespace *ns)
+{
+ struct hlist_node *elem;
+ struct pid *pid;
+
+ hlist_for_each_entry_rcu(pid, elem,
+   &pid_hash[pid_hashfn(nr)], pid_chain) {
+  if (pid->nr == nr)
+   return pid;
+ }
+ return NULL;
+}
+#else
+#endif
+
+EXPORT_SYMBOL_GPL(find_pid_ns);
+
 fastcall void put_pid(struct pid *pid)
 {
  if (!pid)
   return;
  if ((atomic_read(&pid->count) == 1) ||
-      atomic_dec_and_test(&pid->count))
+      atomic_dec_and_test(&pid->count)) {
+  free_pid_nrs(pid);
   kmem_cache_free(pid_cachep, pid);
+ }
 }
 EXPORT_SYMBOL_GPL(put_pid);

@@ -192,14 +243,7 @@ static void delayed_put_pid(struct rcu_h

 fastcall void free_pid(struct pid *pid)
 {
- /* We can be called with write_lock_irq(&tasklist_lock) held */
- unsigned long flags;
-
- spin_lock_irqsave(&pidmap_lock, flags);
```

```
- hlist_del_rcu(&pid->pid_chain);
- spin_unlock_irqrestore(&pidmap_lock, flags);
-
- free_pidmap(&init_pid_ns, pid->nr);
+ unhash_pid_nrs(pid);
  call_rcu(&pid->rcu, delayed_put_pid);
 }

@@ -207,47 +251,25 @@ struct pid *alloc_pid(void)
 {
  struct pid *pid;
  enum pid_type type;
- int nr = -1;

  pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
  if (!pid)
   goto out;

- nr = alloc_pidmap(current->nsproxy->pid_ns);
- if (nr < 0)
-  goto out_free;
-
  atomic_set(&pid->count, 1);
- pid->nr = nr;
  for (type = 0; type < PIDTYPE_MAX; ++type)
   INIT_HLIST_HEAD(&pid->tasks[type]);

- spin_lock_irq(&pidmap_lock);
- hlist_add_head_rcu(&pid->pid_chain, &pid_hash[pid_hashfn(pid->nr)]);
- spin_unlock_irq(&pidmap_lock);
+ if (alloc_pid_nrs(pid))
+  goto out_free;

-out:
  return pid;

 out_free:
  kmem_cache_free(pid_cachep, pid);
- pid = NULL;
- goto out;
-}
-
-struct pid * fastcall find_pid_ns(int nr, struct pid_namespace *ns)
-{
- struct hlist_node *elem;
- struct pid *pid;
-
- hlist_for_each_entry_rcu(pid, elem,
```

```
-   &pid_hash[pid_hashfn(nr)], pid_chain) {
-  if (pid->nr == nr)
-   return pid;
- }
+out:
  return NULL;
 }
-EXPORT_SYMBOL_GPL(find_pid_ns);

 /*
  * attach_pid() must be called with the tasklist_lock write-held.
```