Subject: [PATCH 1/2] Uninline find_task_by_xxx set of functions
Posted by Pavel Emelianov on Mon, 24 Sep 2007 15:20:29 GMT
View Forum Message <> Reply to Message

The find_task_by_something is a set of macros are used to find
task by pid depending on what kind of pid is proposed - global
or virtual one. All of them are wrappers above the most generic
one - find_task_by_pid_type_ns() - and just substitute some
args for it.

It turned out, that dereferencing the current->nsproxy->pid_ns
construction and pushing one more argument on the stack inline
cause kernel text size to grow.

This patch moves all this stuff out-of-line into kernel/pid.c.
Together with the next patch it saves a bit less than 400 bytes
from the .text section.

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

```
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 12ef317..8c0e146 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1495,9 +1495,8 @@ extern struct pid_namespace init_pid_ns;
  *      type and namespace specified
  * find_task_by_pid_ns():
  *      finds a task by its pid in the specified namespace
- * find_task_by_pid_type():
- *      finds a task by its global id with the specified type, e.g.
- *      by global session id
+ * find_task_by_vpid():
+ *      finds a task by its virtual pid
  * find_task_by_pid():
  *      finds a task by its global pid
  *
@@ -1507,12 +1506,10 @@ extern struct pid_namespace init_pid_ns;
 extern struct task_struct *find_task_by_pid_type_ns(int type, int pid,
   struct pid_namespace *ns);

-#define find_task_by_pid_ns(nr, ns) \
-  find_task_by_pid_type_ns(PIDTYPE_PID, nr, ns)
-#define find_task_by_pid_type(type, nr) \
-  find_task_by_pid_type_ns(type, nr, &init_pid_ns)
-#define find_task_by_pid(nr)  \
-  find_task_by_pid_type(PIDTYPE_PID, nr)
```

```
+extern struct task_struct *find_task_by_pid(pid_t nr);
+extern struct task_struct *find_task_by_vpid(pid_t nr);
+extern struct task_struct *find_task_by_pid_ns(pid_t nr,
+  struct pid_namespace *ns);

 extern void __set_special_pids(pid_t session, pid_t pgrp);

diff --git a/kernel/pid.c b/kernel/pid.c
index e2e060e..f029c17 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -368,6 +380,25 @@ struct task_struct *find_task_by_pid_typ

 EXPORT_SYMBOL(find_task_by_pid_type_ns);

+struct task_struct *find_task_by_pid(pid_t nr)
+{
+ return find_task_by_pid_type_ns(PIDTYPE_PID, nr, &init_pid_ns);
+}
+EXPORT_SYMBOL(find_task_by_pid);
+
+struct task_struct *find_task_by_vpid(pid_t vnr)
+{
+ return find_task_by_pid_type_ns(PIDTYPE_PID, vnr,
+   current->nsproxy->pid_ns);
+}
+EXPORT_SYMBOL(find_task_by_vpid);
+
+struct task_struct *find_task_by_pid_ns(pid_t nr, struct pid_namespace *ns)
+{
+ return find_task_by_pid_type_ns(PIDTYPE_PID, nr, ns);
+}
+EXPORT_SYMBOL(find_task_by_pid_ns);
+
 struct pid *get_task_pid(struct task_struct *task, enum pid_type type)
 {
  struct pid *pid;
diff --git a/fs/ioprio.c b/fs/ioprio.c
index 0a615f8..d6ff77e 100644
--- a/fs/ioprio.c
+++ b/fs/ioprio.c
@@ -94,8 +94,7 @@ asmlinkage long sys_ioprio_set(int which
    if (!who)
     p = current;
    else
-    p = find_task_by_pid_ns(who,
-      current->nsproxy->pid_ns);
+    p = find_task_by_vpid(who);
```

```
    if (p)
     ret = set_task_ioprio(p, ioprio);
    break;
@@ -182,8 +181,7 @@ asmlinkage long sys_ioprio_get(int which
    if (!who)
     p = current;
    else
-    p = find_task_by_pid_ns(who,
-      current->nsproxy->pid_ns);
+    p = find_task_by_vpid(who);
    if (p)
     ret = get_task_ioprio(p);
    break;
diff --git a/kernel/capability.c b/kernel/capability.c
index 0440d6d..4a881b8 100644
--- a/kernel/capability.c
+++ b/kernel/capability.c
@@ -63,8 +63,7 @@ asmlinkage long sys_capget(cap_user_head
  read_lock(&tasklist_lock);

  if (pid && pid != task_pid_vnr(current)) {
-  target = find_task_by_pid_ns(pid,
-    current->nsproxy->pid_ns);
+  target = find_task_by_vpid(pid);
   if (!target) {
    ret = -ESRCH;
    goto out;
@@ -198,8 +197,7 @@ asmlinkage long sys_capset(cap_user_head
  read_lock(&tasklist_lock);

  if (pid > 0 && pid != task_pid_vnr(current)) {
-  target = find_task_by_pid_ns(pid,
-    current->nsproxy->pid_ns);
+  target = find_task_by_vpid(pid);
   if (!target) {
    ret = -ESRCH;
    goto out;
diff --git a/kernel/futex.c b/kernel/futex.c
index 0c1b777..0d51412 100644
--- a/kernel/futex.c
+++ b/kernel/futex.c
@@ -444,9 +444,7 @@ static struct task_struct * futex_find_g
  struct task_struct *p;

  rcu_read_lock();
- p = find_task_by_pid_ns(pid,
-   current->nsproxy->pid_ns);
-
```

```
+ p = find_task_by_vpid(pid);
  if (!p || ((current->euid != p->euid) && (current->euid != p->uid)))
   p = ERR_PTR(-ESRCH);
  else
@@ -1856,8 +1854,7 @@ sys_get_robust_list(int pid, struct robu

   ret = -ESRCH;
   rcu_read_lock();
-  p = find_task_by_pid_ns(pid,
-    current->nsproxy->pid_ns);
+  p = find_task_by_vpid(pid);
   if (!p)
    goto err_unlock;
   ret = -EPERM;
diff --git a/kernel/futex_compat.c b/kernel/futex_compat.c
index a21e766..d64f7c3 100644
--- a/kernel/futex_compat.c
+++ b/kernel/futex_compat.c
@@ -118,8 +118,7 @@ compat_sys_get_robust_list(int pid, comp

   ret = -ESRCH;
   read_lock(&tasklist_lock);
-  p = find_task_by_pid_ns(pid,
-    current->nsproxy->pid_ns);
+  p = find_task_by_vpid(pid);
   if (!p)
    goto err_unlock;
   ret = -EPERM;
diff --git a/kernel/ptrace.c b/kernel/ptrace.c
index 812f1f3..593afeb 100644
--- a/kernel/ptrace.c
+++ b/kernel/ptrace.c
@@ -444,8 +444,7 @@ struct task_struct *ptrace_get_task_stru
   return ERR_PTR(-EPERM);

  read_lock(&tasklist_lock);
- child = find_task_by_pid_ns(pid,
-   current->nsproxy->pid_ns);
+ child = find_task_by_vpid(pid);
  if (child)
   get_task_struct(child);

diff --git a/kernel/sched.c b/kernel/sched.c
index 80f5695..ec03e40 100644
--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -4156,8 +4156,7 @@ struct task_struct *idle_task(int cpu)
  */
```

```
 static inline struct task_struct *find_process_by_pid(pid_t pid)
 {
- return pid ?
-  find_task_by_pid_ns(pid, current->nsproxy->pid_ns) : current;
+ return pid ? find_task_by_vpid(pid) : current;
 }

 /* Actually do priority change: must hold rq lock. */
diff --git a/kernel/signal.c b/kernel/signal.c
index c5d72be..c308c2d 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -2237,7 +2237,7 @@ static int do_tkill(int tgid, int pid, i
  info.si_uid = current->uid;

  read_lock(&tasklist_lock);
- p = find_task_by_pid_ns(pid, current->nsproxy->pid_ns);
+ p = find_task_by_vpid(pid);
  if (p && (tgid <= 0 || task_tgid_vnr(p) == tgid)) {
   error = check_kill_permission(sig, &info, p);
   /*
diff --git a/kernel/sys.c b/kernel/sys.c
index 787b73e..796299c 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -151,8 +151,7 @@ asmlinkage long sys_setpriority(int whic
  switch (which) {
   case PRIO_PROCESS:
    if (who)
-    p = find_task_by_pid_ns(who,
-      current->nsproxy->pid_ns);
+    p = find_task_by_vpid(who);
    else
     p = current;
    if (p)
@@ -209,8 +208,7 @@ asmlinkage long sys_getpriority(int whic
  switch (which) {
   case PRIO_PROCESS:
    if (who)
-    p = find_task_by_pid_ns(who,
-      current->nsproxy->pid_ns);
+    p = find_task_by_vpid(who);
    else
     p = current;
    if (p) {
@@ -1065,7 +1063,8 @@ asmlinkage long sys_setsid(void)
   * session id and so the check will always fail and make it so
   * init cannot successfully call setsid.
```

```
   */
- if (session > 1 && find_task_by_pid_type(PIDTYPE_PGID, session))
+ if (session > 1 && find_task_by_pid_type_ns(PIDTYPE_PGID,
+     session, &init_pid_ns))
   goto out;

  group_leader->signal->leader = 1;
diff --git a/mm/mempolicy.c b/mm/mempolicy.c
index a09ca3b..c1592a9 100644
--- a/mm/mempolicy.c
+++ b/mm/mempolicy.c
@@ -941,8 +941,7 @@ asmlinkage long sys_migrate_pages(pid_t

  /* Find the mm_struct */
  read_lock(&tasklist_lock);
- task = pid ?
-  find_task_by_pid_ns(pid, current->nsproxy->pid_ns) : current;
+ task = pid ? find_task_by_vpid(pid) : current;
  if (!task) {
   read_unlock(&tasklist_lock);
   return -ESRCH;
diff --git a/mm/migrate.c b/mm/migrate.c
index 6f7e4f3..7fff95d 100644
--- a/mm/migrate.c
+++ b/mm/migrate.c
@@ -931,8 +931,7 @@ asmlinkage long sys_move_pages(pid_t pid

  /* Find the mm_struct */
  read_lock(&tasklist_lock);
- task = pid ?
-  find_task_by_pid_ns(pid, current->nsproxy->pid_ns) : current;
+ task = pid ? find_task_by_vpid(pid) : current;
  if (!task) {
   read_unlock(&tasklist_lock);
   return -ESRCH;
```