Subject: [PATCH 4/4] - v2 - PID: use the target ID specified in procfs
Posted by Nadia Derbey on Fri, 18 Apr 2008 05:45:03 GMT
View Forum Message <> Reply to Message

[PATCH 04/04]

This patch makes use of the target ids specified by a previous write to
/proc/self/task/<tid>/next_id as the ids to use to allocate the next upid nrs.
Upper levels upid nrs that are not specified in next_pids file are left to the
kernel choice.

Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

---
 include/linux/pid.h | 2
 kernel/fork.c | 3 -
 kernel/pid.c | 141 ++-------
 3 files changed, 126 insertions(+), 20 deletions(-)

Index: linux-2.6.25-rc8-mm2/include/linux/pid.h
===
--- linux-2.6.25-rc8-mm2.orig/include/linux/pid.h 2008-04-17 12:50:22.000000000 +0200
+++ linux-2.6.25-rc8-mm2/include/linux/pid.h 2008-04-17 17:42:11.000000000 +0200
@@ -121,7 +121,7 @@ extern struct pid *find_get_pid(int nr);
 extern struct pid *find_ge_pid(int nr, struct pid_namespace *);
 int next_pidmap(struct pid_namespace *pid_ns, int last);

-extern struct pid *alloc_pid(struct pid_namespace *ns);
+extern struct pid *alloc_pid(struct pid_namespace *ns, int *retval);
 extern void free_pid(struct pid *pid);

 /*
Index: linux-2.6.25-rc8-mm2/kernel/fork.c
===
--- linux-2.6.25-rc8-mm2.orig/kernel/fork.c 2008-04-17 13:49:09.000000000 +0200
+++ linux-2.6.25-rc8-mm2/kernel/fork.c 2008-04-17 17:42:48.000000000 +0200
@@ -1200,8 +1200,7 @@ static struct task_struct *copy_process(
   goto bad_fork_cleanup_io;

  if (pid != &init_struct_pid) {
- retval = -ENOMEM;
- pid = alloc_pid(task_active_pid_ns(p));
+ pid = alloc_pid(task_active_pid_ns(p), &retval);
   if (!pid)
    goto bad_fork_cleanup_io;

Index: linux-2.6.25-rc8-mm2/kernel/pid.c
===

```
--- linux-2.6.25-rc8-mm2.orig/kernel/pid.c 2008-04-17 12:50:25.000000000 +0200
+++ linux-2.6.25-rc8-mm2/kernel/pid.c 2008-04-17 17:50:00.000000000 +0200
@@ -122,6 +122,26 @@ static void free_pidmap(struct upid *upi
  atomic_inc(&map->nr_free);
 }

+static inline int alloc_pidmap_page(struct pidmap *map)
+{
+ if (unlikely(!map->page)) {
+  void *page = kzalloc(PAGE_SIZE, GFP_KERNEL);
+  /*
+   * Free the page if someone raced with us
+   * installing it:
+   */
+  spin_lock_irq(&pidmap_lock);
+  if (map->page)
+   kfree(page);
+  else
+   map->page = page;
+  spin_unlock_irq(&pidmap_lock);
+  if (unlikely(!map->page))
+   return -1;
+ }
+ return 0;
+}
+
 static int alloc_pidmap(struct pid_namespace *pid_ns)
 {
  int i, offset, max_scan, pid, last = pid_ns->last_pid;
@@ -134,21 +154,8 @@ static int alloc_pidmap(struct pid_names
  map = &pid_ns->pidmap[pid/BITS_PER_PAGE];
  max_scan = (pid_max + BITS_PER_PAGE - 1)/BITS_PER_PAGE - !offset;
  for (i = 0; i <= max_scan; ++i) {
-  if (unlikely(!map->page)) {
-   void *page = kzalloc(PAGE_SIZE, GFP_KERNEL);
-   /*
-    * Free the page if someone raced with us
-    * installing it:
-    */
-   spin_lock_irq(&pidmap_lock);
-   if (map->page)
-    kfree(page);
-   else
-    map->page = page;
-   spin_unlock_irq(&pidmap_lock);
-   if (unlikely(!map->page))
-    break;
-  }
```

```
+  if (unlikely(alloc_pidmap_page(map)))
+   break;
   if (likely(atomic_read(&map->nr_free))) {
    do {
     if (!test_and_set_bit(offset, map->page)) {
@@ -182,6 +189,35 @@ static int alloc_pidmap(struct pid_names
  return -1;
 }

+/*
+ * Return a predefined pid value if successful (ID_AT(pid_l, level)),
+ * -errno else
+ */
+static int alloc_fixed_pidmap(struct pid_namespace *pid_ns,
+   struct sys_id *pid_l, int level)
+{
+ int offset, pid;
+ struct pidmap *map;
+
+ pid = ID_AT(pid_l, level);
+ if (pid < RESERVED_PIDS || pid >= pid_max)
+  return -EINVAL;
+
+ map = &pid_ns->pidmap[pid / BITS_PER_PAGE];
+
+ if (unlikely(alloc_pidmap_page(map)))
+  return -ENOMEM;
+
+ offset = pid & BITS_PER_PAGE_MASK;
+ if (test_and_set_bit(offset, map->page))
+  return -EBUSY;
+
+ atomic_dec(&map->nr_free);
+ pid_ns->last_pid = max(pid_ns->last_pid, pid);
+
+ return pid;
+}
+
 int next_pidmap(struct pid_namespace *pid_ns, int last)
 {
  int offset;
@@ -243,20 +279,91 @@ void free_pid(struct pid *pid)
  call_rcu(&pid->rcu, delayed_put_pid);
 }

-struct pid *alloc_pid(struct pid_namespace *ns)
+/*
+ * Called by alloc_pid() to use a list of predefined ids for the calling
```

```
+ * process' upper ns levels.
+ * Returns next pid ns to visit if successful (may be NULL if walked through
+ * the entire pid ns hierarchy).
+ * i is filled with next level to be visited (useful for the error cases).
+ */
+static struct pid_namespace *set_predefined_pids(struct pid_namespace *ns,
+      struct pid *pid,
+      struct sys_id *pid_l,
+      int *next_level)
+{
+ struct pid_namespace *tmp;
+ int rel_level, i, nr;
+
+ rel_level = pid_l->nids - 1;
+ if (rel_level > ns->level)
+  return ERR_PTR(-EINVAL);
+
+ tmp = ns;
+
+ /*
+  * Use the predefined upid nrs for levels ns->level down to
+  * ns->level - rel_level
+  */
+ for (i = ns->level ; rel_level >= 0; i--, rel_level--) {
+  nr = alloc_fixed_pidmap(tmp, pid_l, rel_level);
+  if (nr < 0) {
+   tmp = ERR_PTR(nr);
+   goto out;
+  }
+
+  pid->numbers[i].nr = nr;
+  pid->numbers[i].ns = tmp;
+  tmp = tmp->parent;
+ }
+
+ id_blocks_free(pid_l);
+out:
+ *next_level = i;
+ return tmp;
+}
+
+struct pid *alloc_pid(struct pid_namespace *ns, int *retval)
 {
  struct pid *pid;
  enum pid_type type;
  int i, nr;
  struct pid_namespace *tmp;
  struct upid *upid;
```

```
+ struct sys_id *pid_l;

+ *retval = -ENOMEM;
  pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
  if (!pid)
   goto out;

  tmp = ns;
- for (i = ns->level; i >= 0; i--) {
+ i = ns->level;
+
+ /*
+  * If there is a list of upid nrs specified, use it instead of letting
+  * the kernel chose the upid nrs for us.
+  */
+ pid_l = current->next_id;
+ if (pid_l && pid_l->nids) {
+  /*
+   * returns the next ns to be visited in the following loop
+   * (or NULL if we are done).
+   * i is filled in with the next level to be visited. We need
+   * it to undo things in the error cases.
+   */
+  tmp = set_predefined_pids(ns, pid, pid_l, &i);
+  if (IS_ERR(tmp)) {
+   *retval = PTR_ERR(tmp);
+   goto out_free;
+  }
+  current->next_id = NULL;
+ }
+
+ *retval = -ENOMEM;
+ /*
+  * Let the lower levels upid nrs be automatically allocated
+  */
+ for ( ; i >= 0; i--) {
   nr = alloc_pidmap(tmp);
   if (nr < 0)
    goto out_free;

--
```