Subject: [PATCH 9/16] Make proc_flust_task to flush entries from multiple proc trees
Posted by Pavel Emelianov on Fri, 06 Jul 2007 08:08:09 GMT
View Forum Message <> Reply to Message

Since a task will appear in more than one proc tree we need to shrink many
trees. For this case we pass the struct pid to proc_flush_task() and shrink
the mounts of all the namespaces this pid belongs to.

The NULL passed to it means that only global mount is to be flushed.

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

---

```
 fs/proc/base.c         |   25 ++++++++++++++++++++++---
 include/linux/proc_fs.h |    6 ++++--
 kernel/exit.c          |   18 ++++++++++++++++--
 3 files changed, 43 insertions(+), 6 deletions(-)

diff -upr linux-2.6.22-rc4-mm2.orig/fs/proc/base.c linux-2.6.22-rc4-mm2-2/fs/proc/base.c
--- linux-2.6.22-rc4-mm2.orig/fs/proc/base.c 2007-06-14 12:14:29.000000000 +0400
+++ linux-2.6.22-rc4-mm2-2/fs/proc/base.c 2007-07-04 19:00:38.000000000 +0400
@@ -75,6 +75,7 @@
 #include <linux/nsproxy.h>
 #include <linux/oom.h>
 #include <linux/elf.h>
+#include <linux/pid_namespace.h>
 #include "internal.h"

 /* NOTE:
@@ -2183,7 +2184,7 @@ static const struct inode_operations pro
  *       that no dcache entries will exist at process exit time it
  *       just makes it very unlikely that any will persist.
  */
-void proc_flush_task(struct task_struct *task)
+static void proc_flush_task_mnt(struct task_struct *task, struct vfsmount *mnt)
 {
  struct dentry *dentry, *leader, *dir;
  char buf[PROC_NUMBUF];
@@ -2191,7 +2192,7 @@ void proc_flush_task(struct task_struct

  name.name = buf;
  name.len = snprintf(buf, sizeof(buf), "%d", task->pid);
- dentry = d_hash_and_lookup(proc_mnt->mnt_root, &name);
+ dentry = d_hash_and_lookup(mnt->mnt_root, &name);
  if (dentry) {
   shrink_dcache_parent(dentry);
```

```
   d_drop(dentry);
@@ -2203,7 +2204,7 @@ void proc_flush_task(struct task_struct

  name.name = buf;
  name.len = snprintf(buf, sizeof(buf), "%d", task->tgid);
- leader = d_hash_and_lookup(proc_mnt->mnt_root, &name);
+ leader = d_hash_and_lookup(mnt->mnt_root, &name);
  if (!leader)
   goto out;

@@ -2229,6 +2230,24 @@ out:
  return;
 }

+/*
+ * when flushing dentries from proc one need to flush them from global
+ * proc (proc_mnt) and from all the namespaces' procs this task was seen
+ * in. this call is supposed to make all this job.
+ */
+
+void proc_flush_task(struct task_struct *task, struct pid *pid)
+{
+ int i;
+
+ proc_flush_task_mnt(task, proc_mnt);
+ if (pid == NULL)
+  return;
+
+ for (i = 1; i <= pid->level; i++)
+  proc_flush_task_mnt(task, pid->numbers[i].ns->proc_mnt);
+}
+
 static struct dentry *proc_pid_instantiate(struct inode *dir,
        struct dentry * dentry,
        struct task_struct *task, const void *ptr)
diff -upr linux-2.6.22-rc4-mm2.orig/include/linux/proc_fs.h
linux-2.6.22-rc4-mm2-2/include/linux/proc_fs.h
--- linux-2.6.22-rc4-mm2.orig/include/linux/proc_fs.h 2007-06-14 12:14:29.000000000 +0400
+++ linux-2.6.22-rc4-mm2-2/include/linux/proc_fs.h 2007-07-04 19:00:38.000000000 +0400
@@ -111,7 +111,7 @@ extern void proc_misc_init(void);

 struct mm_struct;

-void proc_flush_task(struct task_struct *task);
+void proc_flush_task(struct task_struct *task, struct pid *pid);
 struct dentry *proc_pid_lookup(struct inode *dir, struct dentry * dentry, struct nameidata *);
 int proc_pid_readdir(struct file * filp, void * dirent, filldir_t filldir);
 unsigned long task_vsize(struct mm_struct *);
```

```
@@ -223,7 +227,9 @@ static inline void proc_net_remove(const
 #define proc_net_create(name, mode, info) ({ (void)(mode), NULL; })
 static inline void proc_net_remove(const char *name) {}

-static inline void proc_flush_task(struct task_struct *task) { }
+static inline void proc_flush_task(struct task_struct *task, struct pid *pid)
+{
+}

 static inline struct proc_dir_entry *create_proc_entry(const char *name,
  mode_t mode, struct proc_dir_entry *parent) { return NULL; }
diff -upr linux-2.6.22-rc4-mm2.orig/kernel/exit.c linux-2.6.22-rc4-mm2-2/kernel/exit.c
--- linux-2.6.22-rc4-mm2.orig/kernel/exit.c 2007-07-04 19:00:38.000000000 +0400
+++ linux-2.6.22-rc4-mm2-2/kernel/exit.c 2007-07-04 19:00:38.000000000 +0400
@@ -154,6 +154,7 @@ static void delayed_put_task_struct(stru

 void release_task(struct task_struct * p)
 {
+ struct pid *pid;
  struct task_struct *leader;
  int zap_leader;
 repeat:
@@ -161,6 +162,20 @@ repeat:
  write_lock_irq(&tasklist_lock);
  ptrace_unlink(p);
  BUG_ON(!list_empty(&p->ptrace_list) || !list_empty(&p->ptrace_children));
+ /*
+  * we have to keep this pid till proc_flush_task() to make
+  * it possible to flush all dentries holding it. pid will
+  * be put ibidem
+  *
+  * however if the pid belogs to init namespace only, we can
+  * optimize this out
+  */
+ pid = task_pid(p);
+ if (!pid_ns_accessible(&init_pid_ns, pid))
+  get_pid(pid);
+ else
+  pid = NULL;
+
  __exit_signal(p);

  /*
@@ -185,7 +200,8 @@ repeat:
  }

  write_unlock_irq(&tasklist_lock);
- proc_flush_task(p);
```

```
+ proc_flush_task(p, pid);
+ put_pid(pid);
  release_thread(p);
  call_rcu(&p->rcu, delayed_put_task_struct);
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers