Subject: [PATCH 17/17] Pid-NS(V3) Introduce proc_mnt for pid_ns
Posted by Sukadev Bhattiprolu on Sat, 16 Jun 2007 23:05:55 GMT
View Forum Message <> Reply to Message

From: sukadev@linux.vnet.ibm.com
Subject: [PATCH 17/17] Pid-NS(V3) Introduce proc_mnt for pid_ns

The following patch completes the removal of the global proc_mnt.
It fetches the mnt on which to do dentry invalidations from the
pid_namespace in which the task appears.

For now, there is only one pid namespace in mainline so this is
straightforward. In the -lxc tree we'll have to do something
more complex. The proc_flush_task() code takes a task, and
needs to be able to find the corresponding proc superblocks on
which that tasks's /proc/<pid> directories could appear. We
can tell in which pid namespaces a task appears, so I put a
pointer from the pid namespace to the corresponding proc_mnt.

/proc currently has some special code to make sure that the root
directory gets set up correctly. It proc_mnt variable in order
to find its way to the root inode.

Changelog:

2.6.22-rc4-mm2-pidns1:

- Call proc_fill_super once per pid namespace
- Call proc_flush_task() before detaching a task's 'struct pid'.
- Get a reference to pid namespace when mounting/remounting /proc.
  Put this reference when unmounting.

Signed-off-by: Dave Hansen <haveblue@us.ibm.com>
Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

```
 fs/proc/base.c              |   30 +++++++++++++++++++-
 fs/proc/inode.c             |   12 ++++++-
 fs/proc/root.c              |   65 +++++++++++++++++++++++++++++++------------
 include/linux/pid_namespace.h |    1
 include/linux/proc_fs.h     |    1
 kernel/exit.c               |    2 -
 6 files changed, 87 insertions(+), 24 deletions(-)
```

Index: lx26-22-rc4-mm2/fs/proc/base.c
==========
--- lx26-22-rc4-mm2.orig/fs/proc/base.c 2007-06-16 02:37:15.000000000 -0700
+++ lx26-22-rc4-mm2/fs/proc/base.c 2007-06-16 04:15:23.000000000 -0700

```
@@ -2164,9 +2164,11 @@ static const struct inode_operations pro
 };

 /**
- * proc_flush_task -  Remove dcache entries for @task from the /proc dcache.
+ * proc_flush_task_from_pid_ns -  Remove dcache entries for @task
+ *       from the /proc dcache.
  *
  * @task: task that should be flushed.
+ * @pid_ns: pid_namespace in which that task appears
  *
  * Looks in the dcache for
  * /proc/@pid
@@ -2184,11 +2186,20 @@ 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_from_pid_ns(struct task_struct *task,
+    struct pid_namespace* pid_ns)
 {
  struct dentry *dentry, *leader, *dir;
  char buf[PROC_NUMBUF];
  struct qstr name;
+ struct vfsmount *proc_mnt = pid_ns->proc_mnt;
+
+ /*
+  * It is possible that no /procs have been instantiated
+  * for this particular pid namespace.
+  */
+ if (!proc_mnt || !proc_mnt->mnt_root)
+  return;

  name.name = buf;
  name.len = snprintf(buf, sizeof(buf), "%d", task->pid);
@@ -2230,6 +2241,21 @@ out:
  return;
 }

+void proc_flush_task(struct task_struct *task)
+{
+ int i;
+ struct pid *pid;
+ struct upid* upid;
+
+ pid = task_pid(task);
+ if (!pid)
+  return;
```

```
+
+ upid = &pid->upid_list[0];
+ for (i = 0; i < pid->num_upids; i++, upid++)
+  proc_flush_task_from_pid_ns(task, upid->pid_ns);
+}
+
 static struct dentry *proc_pid_instantiate(struct inode *dir,
         struct dentry * dentry,
         struct task_struct *task, const void *ptr)
Index: lx26-22-rc4-mm2/fs/proc/inode.c
===================================================================
--- lx26-22-rc4-mm2.orig/fs/proc/inode.c 2007-06-16 02:37:15.000000000 -0700
+++ lx26-22-rc4-mm2/fs/proc/inode.c 2007-06-16 04:15:23.000000000 -0700
@@ -6,6 +6,7 @@

 #include <linux/time.h>
 #include <linux/proc_fs.h>
+#include <linux/hardirq.h>
 #include <linux/kernel.h>
 #include <linux/mm.h>
 #include <linux/string.h>
@@ -19,6 +20,7 @@

 #include <asm/system.h>
 #include <asm/uaccess.h>
+#include <linux/pid_namespace.h>

 #include "internal.h"

@@ -75,8 +77,6 @@ static void proc_delete_inode(struct ino
  clear_inode(inode);
 }

-struct vfsmount *proc_mnt;
-
 static void proc_read_inode(struct inode * inode)
 {
  inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;
@@ -431,6 +431,8 @@ out_mod:

 int proc_fill_super(struct super_block *s, void *data, int silent)
 {
+ struct pid_namespace *pid_ns = data;
+ struct proc_inode *ei;
  struct inode * root_inode;

  s->s_flags |= MS_NODIRATIME | MS_NOSUID | MS_NOEXEC;
@@ -439,6 +441,7 @@ int proc_fill_super(struct super_block *
```

```
  s->s_magic = PROC_SUPER_MAGIC;
  s->s_op = &proc_sops;
  s->s_time_gran = 1;
+ s->s_fs_info = get_pid_ns(pid_ns);

  de_get(&proc_root);
  root_inode = proc_get_inode(s, PROC_ROOT_INO, &proc_root);
@@ -449,6 +452,11 @@ int proc_fill_super(struct super_block *
  s->s_root = d_alloc_root(root_inode);
  if (!s->s_root)
   goto out_no_root;
+ /* Seed the root directory with a pid so it doesn't need
+  * to be special in base.c.
+  */
+ ei = PROC_I(root_inode);
+ ei->pid = find_get_pid(1);
  return 0;

 out_no_root:
Index: lx26-22-rc4-mm2/fs/proc/root.c
===================================================================
--- lx26-22-rc4-mm2.orig/fs/proc/root.c 2007-06-16 04:15:23.000000000 -0700
+++ lx26-22-rc4-mm2/fs/proc/root.c 2007-06-16 04:15:23.000000000 -0700
@@ -12,38 +12,73 @@
 #include <linux/time.h>
 #include <linux/proc_fs.h>
 #include <linux/stat.h>
+#include <linux/hardirq.h>
 #include <linux/init.h>
 #include <linux/sched.h>
 #include <linux/module.h>
 #include <linux/bitops.h>
 #include <linux/smp_lock.h>
 #include <linux/mount.h>
+#include <linux/pid_namespace.h>

 #include "internal.h"

 struct proc_dir_entry *proc_net, *proc_net_stat, *proc_bus, *proc_root_fs, *proc_root_driver;

+static int proc_test_sb(struct super_block *s, void *data)
+{
+ struct pid_namespace *pid_ns = data;
+ if (s->s_fs_info == pid_ns)
+  return 1;
+ return 0;
+}
+
```

```
 static int proc_get_sb(struct file_system_type *fs_type,
  int flags, const char *dev_name, void *data, struct vfsmount *mnt)
 {
- if (proc_mnt) {
-  /* Seed the root directory with a pid so it doesn't need
-   * to be special in base.c.  I would do this earlier but
-   * the only task alive when /proc is mounted the first time
-   * is the init_task and it doesn't have any pids.
-   */
-  struct proc_inode *ei;
-  ei = PROC_I(proc_mnt->mnt_sb->s_root->d_inode);
-  if (!ei->pid)
-   ei->pid = find_get_pid(1);
+ int error;
+ struct super_block *s;
+ struct pid_namespace *pid_ns;
+
+ /*
+  * We can eventually derive this out of whatever mount
+  * arguments the user supplies, but just take it from
+  * current for now.
+  */
+ pid_ns = task_active_pid_ns(current);
+
+ s = sget(fs_type, proc_test_sb, set_anon_super, pid_ns);
+ if (IS_ERR(s))
+  return PTR_ERR(s);
+
+ if (!s->s_root) {
+  error = proc_fill_super(s, pid_ns, 0);
+  if (error) {
+   deactivate_super(s);
+   return error;
+  }
  }
- return get_sb_single(fs_type, flags, data, proc_fill_super, mnt);
+
+ if (!pid_ns->proc_mnt)
+  pid_ns->proc_mnt = mnt;
+
+ do_remount_sb(s, flags, data, 0);
+ return simple_set_mnt(mnt, s);
+}
+
+static void proc_kill_sb(struct super_block *s)
+{
+ struct pid_namespace *pid_ns = s->s_fs_info;
+
```

```
+ pid_ns->proc_mnt = NULL;
+ put_pid_ns(pid_ns);
+ kill_anon_super(s);
 }

 static struct file_system_type proc_fs_type = {
  .name  = "proc",
  .get_sb  = proc_get_sb,
- .kill_sb = kill_anon_super,
+ .kill_sb = proc_kill_sb,
  .fs_supers      = LIST_HEAD_INIT(proc_fs_type.fs_supers),
 };

@@ -55,12 +90,6 @@ void __init proc_root_init(void)
  err = register_filesystem(&proc_fs_type);
  if (err)
   return;
- proc_mnt = kern_mount(&proc_fs_type);
- err = PTR_ERR(proc_mnt);
- if (IS_ERR(proc_mnt)) {
-  unregister_filesystem(&proc_fs_type);
-  return;
- }
  proc_misc_init();
  proc_net = proc_mkdir("net", NULL);
  proc_net_stat = proc_mkdir("net/stat", NULL);
Index: lx26-22-rc4-mm2/include/linux/pid_namespace.h
===================================================================
--- lx26-22-rc4-mm2.orig/include/linux/pid_namespace.h 2007-06-16 04:15:23.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/pid_namespace.h 2007-06-16 04:15:23.000000000 -0700
@@ -28,6 +28,7 @@ struct pid_namespace {
  int last_pid;
  struct task_struct *child_reaper;
  atomic_t terminating;
+ struct vfsmount *proc_mnt;
 };

 extern struct pid_namespace init_pid_ns;
Index: lx26-22-rc4-mm2/include/linux/proc_fs.h
===================================================================
--- lx26-22-rc4-mm2.orig/include/linux/proc_fs.h 2007-06-16 02:37:15.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/proc_fs.h 2007-06-16 04:15:23.000000000 -0700
@@ -125,7 +125,6 @@ extern struct proc_dir_entry *create_pro
       struct proc_dir_entry *parent);
 extern void remove_proc_entry(const char *name, struct proc_dir_entry *parent);

-extern struct vfsmount *proc_mnt;
 extern int proc_fill_super(struct super_block *,void *,int);
```

```
 extern struct inode *proc_get_inode(struct super_block *, unsigned int, struct proc_dir_entry *);

Index: lx26-22-rc4-mm2/kernel/exit.c
===================================================================
--- lx26-22-rc4-mm2.orig/kernel/exit.c 2007-06-16 04:15:23.000000000 -0700
+++ lx26-22-rc4-mm2/kernel/exit.c 2007-06-16 04:15:23.000000000 -0700
@@ -157,6 +157,7 @@ void release_task(struct task_struct * p
  struct task_struct *leader;
  int zap_leader;
 repeat:
+ proc_flush_task(p);
  atomic_dec(&p->user->processes);
  write_lock_irq(&tasklist_lock);
  ptrace_unlink(p);
@@ -185,7 +186,6 @@ repeat:
  }

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

---

Subject: Re: [PATCH 17/17] Pid-NS(V3) Introduce proc_mnt for pid_ns
Posted by Pavel Emelianov on Mon, 18 Jun 2007 08:56:56 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> From: sukadev@linux.vnet.ibm.com
> Subject: [PATCH 17/17] Pid-NS(V3) Introduce proc_mnt for pid_ns
>
> The following patch completes the removal of the global proc_mnt.
> It fetches the mnt on which to do dentry invalidations from the
> pid_namespace in which the task appears.
>
> For now, there is only one pid namespace in mainline so this is
> straightforward. In the -lxc tree we'll have to do something
> more complex. The proc_flush_task() code takes a task, and
> needs to be able to find the corresponding proc superblocks on
> which that tasks's /proc/<pid> directories could appear. We
> can tell in which pid namespaces a task appears, so I put a

> pointer from the pid namespace to the corresponding proc_mnt.
>
> /proc currently has some special code to make sure that the root
> directory gets set up correctly. It proc_mnt variable in order
> to find its way to the root inode.
>
> Changelog:
>
> 2.6.22-rc4-mm2-pidns1:
>
> - Call proc_fill_super once per pid namespace
> - Call proc_flush_task() before detaching a task's 'struct pid'.
> - Get a reference to pid namespace when mounting/remounting /proc.
> Put this reference when unmounting.
>
> Signed-off-by: Dave Hansen <haveblue@us.ibm.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
>
> fs/proc/base.c | 30 ++++++++++++++++++-
> fs/proc/inode.c | 12 ++++++-
> fs/proc/root.c | 65 ++++++++++++++++++++++++++++++----------
> include/linux/pid_namespace.h | 1
> include/linux/proc_fs.h | 1
> kernel/exit.c | 2 -
> 6 files changed, 87 insertions(+), 24 deletions(-)
>
[snip]

> @@ -55,12 +90,6 @@ void __init proc_root_init(void)
> err = register_filesystem(&proc_fs_type);
> if (err)
> return;
> - proc_mnt = kern_mount(&proc_fs_type);
> - err = PTR_ERR(proc_mnt);
> - if (IS_ERR(proc_mnt)) {
> - unregister_filesystem(&proc_fs_type);
> - return;
> - }

Wow! Is this safe? Everyone expexts proc_mnt to be not NULL.

> proc_misc_init();
> proc_net = proc_mkdir("net", NULL);
> proc_net_stat = proc_mkdir("net/stat", NULL);

[snip]

> Index: lx26-22-rc4-mm2/kernel/exit.c
> ===
> --- lx26-22-rc4-mm2.orig/kernel/exit.c 2007-06-16 04:15:23.000000000 -0700
> +++ lx26-22-rc4-mm2/kernel/exit.c 2007-06-16 04:15:23.000000000 -0700
> @@ -157,6 +157,7 @@ void release_task(struct task_struct * p
> struct task_struct *leader;
> int zap_leader;
> repeat:
> + proc_flush_task(p);

Such a flush won't actually remove *any* dentries from the tree and thus
may be omitted.

> atomic_dec(&p->user->processes);
> write_lock_irq(&tasklist_lock);
> ptrace_unlink(p);
> @@ -185,7 +186,6 @@ repeat:
> }
>
> write_unlock_irq(&tasklist_lock);
> - proc_flush_task(p);
> 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

---

Subject: Re: [PATCH 17/17] Pid-NS(V3) Introduce proc_mnt for pid_ns
Posted by Sukadev Bhattiprolu on Wed, 20 Jun 2007 01:19:11 GMT
View Forum Message <> Reply to Message

Pavel Emelianov [xemul@openvz.org] wrote:
| sukadev@us.ibm.com wrote:
| > From: sukadev@linux.vnet.ibm.com
| > Subject: [PATCH 17/17] Pid-NS(V3) Introduce proc_mnt for pid_ns
| >
| > The following patch completes the removal of the global proc_mnt.
| > It fetches the mnt on which to do dentry invalidations from the
| > pid_namespace in which the task appears.
| >
| > For now, there is only one pid namespace in mainline so this is
| > straightforward. In the -lxc tree we'll have to do something
| > more complex. The proc_flush_task() code takes a task, and

| > needs to be able to find the corresponding proc superblocks on
| > which that tasks's /proc/<pid> directories could appear. We
| > can tell in which pid namespaces a task appears, so I put a
| > pointer from the pid namespace to the corresponding proc_mnt.
| >
| > /proc currently has some special code to make sure that the root
| > directory gets set up correctly. It proc_mnt variable in order
| > to find its way to the root inode.
| >
| > Changelog:
| >
| > 2.6.22-rc4-mm2-pidns1:
| >
| > - Call proc_fill_super once per pid namespace
| > - Call proc_flush_task() before detaching a task's 'struct pid'.
| > - Get a reference to pid namespace when mounting/remounting /proc.
| >   Put this reference when unmounting.
| >
| > Signed-off-by: Dave Hansen <haveblue@us.ibm.com>
| > Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| > ---
| >
| > fs/proc/base.c | 30 ++++++++++++++++++-
| > fs/proc/inode.c | 12 ++++++-
| > fs/proc/root.c | 65 ++++++++++++++++++

|
| >   proc_misc_init();
| >   proc_net = proc_mkdir("net", NULL);
| >   proc_net_stat = proc_mkdir("net/stat", NULL);
|
| [snip]
|
| > Index: lx26-22-rc4-mm2/kernel/exit.c
| > ===
| > --- lx26-22-rc4-mm2.orig/kernel/exit.c 2007-06-16 04:15:23.000000000 -0700
| > +++ lx26-22-rc4-mm2/kernel/exit.c 2007-06-16 04:15:23.000000000 -0700
| > @@ -157,6 +157,7 @@ void release_task(struct task_struct * p
| >   struct task_struct *leader;
| >   int zap_leader;
| > repeat:
| > + proc_flush_task(p);
|
| Such a flush won't actually remove *any* dentries from the tree and thus
| may be omitted.

Again, can you elaborate ?

The previous version had proc_flush_task() much later in release_task() and
that did not free dentries bc task_pid() was NULL. Moving proc_flush_task()
up, should fix that and it did fix the leak we were seeing before.

|
| >   atomic_dec(&p->user->processes);
| >   write_lock_irq(&tasklist_lock);
| >   ptrace_unlink(p);
| > @@ -185,7 +186,6 @@ repeat:
| >   }
| >
| >   write_unlock_irq(&tasklist_lock);
| > - proc_flush_task(p);
| >   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