
Subject: [RFC][PATCH 1/2] remove proc_mnt's use or killing inodes

Posted by [Dave Hansen](#) on Tue, 20 Feb 2007 19:00:55 GMT

[View Forum Message](#) <> [Reply to Message](#)

We use proc_mnt as a shortcut to find a superblock on which to go killing /proc inodes. This will break if we ever have more than one /proc superblock. So, use the superblock list to go find each of the /proc sb's and kill inodes on each superblock.

This does introduce an extra lock grab from what was there before, but the list should be only 1 long 99% of the time, and we don't exactly remove proc entries in hot paths. Note that this *isn't* the path that we use to get rid of the actual /proc pid entries. Those are a different beast.

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

lxc-dave/fs/proc/generic.c | 28 ++++++-----

1 file changed, 23 insertions(+), 5 deletions(-)

diff -puN fs/proc/generic.c~A2-remove-proc_mnt-0 fs/proc/generic.c

--- lxc/fs/proc/generic.c~A2-remove-proc_mnt-0 2007-02-20 10:07:28.000000000 -0800

+++ lxc-dave/fs/proc/generic.c 2007-02-20 10:07:28.000000000 -0800

@@ -597,13 +597,10 @@ static int proc_register(struct proc_dir
 return 0;
}

*/

- * Kill an inode that got unregistered..

- */

-static void proc_kill_inodes(struct proc_dir_entry *de)
+static void proc_kill_inodes_sb(struct proc_dir_entry *de,
+ struct super_block *sb)

{
 struct list_head *p;
- struct super_block *sb = proc_mnt->mnt_sb;

/*

* Actually it's a partial revoke().

@@ -627,6 +624,27 @@ static void proc_kill_inodes(struct proc
 file_list_unlock();
}

*/

+ * Kill an inode that got unregistered..

+ */

```

+static void proc_kill_inodes(struct proc_dir_entry *de)
+{
+ struct list_head *l;
+ struct file_system_type *procfs;
+
+ procfs = get_fs_type("proc");
+ if (!procfs)
+ return;
+
+ spin_lock(&sb_lock);
+ list_for_each(l, &procfs->fs_supers) {
+ struct super_block *sb;
+ sb = list_entry(l, struct super_block, s_instances);
+ proc_kill_inodes_sb(de, sb);
+ }
+ spin_unlock(&sb_lock);
+}
+
+static struct proc_dir_entry *proc_create(struct proc_dir_entry **parent,
+    const char *name,
+    mode_t mode,

```

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>

Subject: [RFC][PATCH 2/2] introduce proc_mnt for pid_ns
Posted by [Dave Hansen](#) on Tue, 20 Feb 2007 19:00:56 GMT
[View Forum Message](#) <> [Reply to Message](#)

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.

Note that the new proc_compare_super() enforces the "one proc sb
per pid_namespace" limit.

/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.

Now that we don't have the global proc_mnt, we can fill in the

root inode's data in `proc_fill_super()`, where it takes a wee bit less work than in `proc_get_sb()`.

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

```
lxc-dave/fs/proc/base.c      | 32 ++++++++
lxc-dave/fs/proc/inode.c     | 10 +++++-
lxc-dave/fs/proc/root.c      | 46 ++++++++-----
lxc-dave/include/linux/pid_namespace.h | 1
lxc-dave/include/linux/proc_fs.h | 1
5 files changed, 67 insertions(+), 23 deletions(-)
```

```
diff -puN fs/proc/base.c~A3-remove-proc_mnt-1 fs/proc/base.c
--- lxc/fs/proc/base.c~A3-remove-proc_mnt-1 2007-02-20 10:07:28.000000000 -0800
+++ lxc-dave/fs/proc/base.c 2007-02-20 10:07:29.000000000 -0800
@@ -71,6 +71,7 @@
#include <linux/seccomp.h>
#include <linux/cpuset.h>
#include <linux/audit.h>
+#include <linux/pid_namespace.h>
#include <linux/poll.h>
#include <linux/nsproxy.h>
#include <linux/oom.h>
@@ -2001,9 +2002,11 @@ 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
@@ -2021,11 +2024,23 @@ 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)
+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;
+
+ /*
+ * It is possible that no /procs have been instantiated
+ * for this particular pid namespace, or that we're trying
+ * to flush for a task without a pid_ns, like a kernel
+ * thread.
+ */
+ if (!pid_ns || !pid_ns->proc_mnt)
+ return;
+ proc_mnt = pid_ns->proc_mnt;

    name.name = buf;
    name.len = snprintf(buf, sizeof(buf), "%d", task->pid);
@@ -2067,6 +2082,19 @@ out:
    return;
}

+void proc_flush_task(struct task_struct *task)
+{
+ struct pid_namespace *pid_ns = NULL;
+ /*
+ * Right now, tasks only appear in their own pid_ns.
+ * With containers this function will change to a loop
+ * over all pid_ns's in which the task appears.
+ */
+ if (task->nsproxy)
+ pid_ns = task->nsproxy->pid_ns;
+ proc_flush_task_from_pid_ns(task, pid_ns);
+}
+
static struct dentry *proc_pid_instantiate(struct inode *dir,
    struct dentry *dentry,
    struct task_struct *task, void *ptr)
diff -puN fs/proc/inode.c~A3-remove-proc_mnt-1 fs/proc/inode.c
--- lxc/fs/proc/inode.c~A3-remove-proc_mnt-1 2007-02-20 10:07:28.000000000 -0800
+++ lxc-dave/fs/proc/inode.c 2007-02-20 10:07:29.000000000 -0800
@@ -67,8 +67,6 @@ 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;
@@ -184,6 +182,8 @@ 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;
@@ -192,6 +192,7 @@ 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 = pid_ns;

  root_inode = proc_get_inode(s, PROC_ROOT_INO, &proc_root);
  if (!root_inode)
@@ -201,6 +202,11 @@ 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:
diff -puN fs/proc/root.c~A3-remove-proc_mnt-1 fs/proc/root.c
--- lxc/fs/proc/root.c~A3-remove-proc_mnt-1 2007-02-20 10:07:28.000000000 -0800
+++ lxc-dave/fs/proc/root.c 2007-02-20 10:07:29.000000000 -0800
@@ -18,26 +18,42 @@
#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_set_sb(struct super_block *s, void *data)
+{
+ int ret;
+ ret = set_anon_super(s, NULL);
+ if (!ret)
+ ret = proc_fill_super(s, data, 0);
+ return ret;
+}
+
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);
- }
- return get_sb_single(fs_type, flags, data, proc_fill_super, mnt);
+ struct super_block *s;
+
+ s = sget(fs_type, proc_test_sb, proc_set_sb, current->nsproxy->pid_ns);
+ if (IS_ERR(s))
+ return PTR_ERR(s);
+ if (!current->nsproxy->pid_ns->proc_mnt)
+ current->nsproxy->pid_ns->proc_mnt = mnt;
+
+ do_remount_sb(s, flags, data, 0);
+ return simple_set_mnt(mnt, s);
}

static struct file_system_type proc_fs_type = {
@@ -54,12 +70,6 @@ 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);
diff -puN include/linux/pid_namespace.h~A3-remove-proc_mnt-1 include/linux/pid_namespace.h
--- lxc/include/linux/pid_namespace.h~A3-remove-proc_mnt-1 2007-02-20 10:07:28.000000000
-0800
+++ lxc-dave/include/linux/pid_namespace.h 2007-02-20 10:07:29.000000000 -0800
@@ -20,6 +20,7 @@ struct pid_namespace {
    struct pidmap pidmap[PIDMAP_ENTRIES];
    int last_pid;
    struct task_struct *child_reaper;
+ struct vfsmount *proc_mnt;
};

extern struct pid_namespace init_pid_ns;
diff -puN include/linux/proc_fs.h~A3-remove-proc_mnt-1 include/linux/proc_fs.h
--- lxc/include/linux/proc_fs.h~A3-remove-proc_mnt-1 2007-02-20 10:07:28.000000000 -0800
+++ lxc-dave/include/linux/proc_fs.h 2007-02-20 10:07:29.000000000 -0800
@@ -129,7 +129,6 @@ 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 *);

```

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>

Subject: Re: [RFC][PATCH 1/2] remove proc_mnt's use or killing inodes
Posted by [ebiederm](#) on Tue, 20 Feb 2007 20:08:48 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dave Hansen <hansendc@us.ibm.com> writes:

```

> We use proc_mnt as a shortcut to find a superblock on which
> to go killing /proc inodes. This will break if we ever have
> more than one /proc superblock. So, use the superblock list
> to go find each of the /proc sb's and kill inodes on each
> superblock.
>
> This does introduce an extra lock grab from what was there
> before, but the list should be only 1 long 99% of the time,
> and we don't exactly remove proc entries in hot paths. Note
> that this *isn't* the path that we use to get rid of the

```

> actual /proc pid entries. Those are a different beast.

Looks good.

The only other thing we could do is turn proc/generic.c into its own filesystem that we mount with magic mounts like nfs does when crossing mount points.

I think that would give us similar problems with killing inodes when a proc entry is removed.

Eric

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>

Subject: Re: [RFC][PATCH 2/2] introduce proc_mnt for pid_ns
Posted by [ebiederm](#) on Tue, 20 Feb 2007 20:31:06 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dave Hansen <hansendc@us.ibm.com> writes:

> 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.
>
> Note that the new proc_compare_super() enforces the "one proc sb
> per pid_namespace" limit.
>
> /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.
>
> Now that we don't have the global proc_mnt, we can fill in the
> root inode's data in proc_fill_super(), where it takes a wee bit
> less work than in proc_get_sb().

Nothing jumps out at me as an obvious problem.

Eric

Containers mailing list

Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
