Subject: [RFC][PATCH 06/14] Populate pid_nrs list with entry for init-pid-ns
Posted by Sukadev Bhattiprolu on Wed, 21 Mar 2007 03:20:04 GMT
View Forum Message <> Reply to Message

From: Cedric Le Goater <clg@fr.ibm.com>
Subject: [RFC][PATCH 06/14] Populate pid_nrs list with entry for init-pid-ns

Create/destroy the pid->pid_nrs list - when allocating/freeing a struct pid.
The pid_nrs list contains just a single struct pid_nr for now, (corresponding
to init-pid-ns).

To enable finding a process based on any of its pid_t values, replace the
use struct pid_nr, rather than struct pid, in the pid_hash table.

Finally, reimplement find_pid() and pid_nr() based on the pid_nrs list.

Changelog:
- [Serge Hallyn's comment]: Add comments on what pid->lock protects
  and that pid->nr will eventually go away.
- [Eric Biederman and containers list comments]: Reworked patches
  to drop support for unsharing pid namespace and to replace
  struct pid with struct pid_nr in the pid_hash table.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

```
---
 include/linux/pid.h |    9 ------
 kernel/pid.c        |   73 +++++++++++++++++++++++++++++++++++++++++++++------------------
 2 files changed, 50 insertions(+), 32 deletions(-)

Index: 2.6.21-rc3-mm2/kernel/pid.c
===================================================================
--- 2.6.21-rc3-mm2.orig/kernel/pid.c
+++ 2.6.21-rc3-mm2/kernel/pid.c
@@ -194,8 +194,15 @@ static void delayed_put_pid(struct rcu_h

 void free_pid_nr(struct pid_nr *pid_nr)
 {
+ /* We can be called with write_lock_irq(&tasklist_lock) held */
+ unsigned long flags;
+
  free_pidmap(pid_nr->pid_ns, pid_nr->nr);

+ spin_lock_irqsave(&pidmap_lock, flags);
+ hlist_del_rcu(&pid_nr->pid_chain);
+ spin_unlock_irqrestore(&pidmap_lock, flags);
+
  hlist_del_init(&pid_nr->node);
```

```
  put_pid_ns(pid_nr->pid_ns);
@@ -204,14 +211,12 @@ void free_pid_nr(struct pid_nr *pid_nr)

 fastcall void free_pid(struct pid *pid)
 {
- /* We can be called with write_lock_irq(&tasklist_lock) held */
- unsigned long flags;
+ struct pid_nr *pid_nr;
+ struct hlist_node *pos, *next;

- spin_lock_irqsave(&pidmap_lock, flags);
- hlist_del_rcu(&pid->pid_chain);
- spin_unlock_irqrestore(&pidmap_lock, flags);
+ hlist_for_each_entry_safe(pid_nr, pos, next, &pid->pid_nrs, node)
+  free_pid_nr(pid_nr);

- free_pidmap(&init_pid_ns, pid->nr);
  call_rcu(&pid->rcu, delayed_put_pid);
 }

@@ -242,50 +247,70 @@ struct pid_nr *alloc_pid_nr(struct pid_n
   */
  pid_nr->pid = pid;

+ spin_lock_irq(&pidmap_lock);
+ hlist_add_head_rcu(&pid_nr->pid_chain, &pid_hash[pid_hashfn(nr)]);
+ spin_unlock_irq(&pidmap_lock);
+
  return pid_nr;
 }

+pid_t pid_nr(struct pid *pid)
+{
+ struct pid_nr* pid_nr;
+ struct hlist_node *pos;
+ struct pid_namespace *ns = task_pid_ns(current);
+
+ if (!pid)
+  return 0;
+
+ hlist_for_each_entry(pid_nr, pos, &pid->pid_nrs, node)
+  if (pid_nr->pid_ns == ns)
+   return pid_nr->nr;
+ return 0;
+}
+EXPORT_SYMBOL_GPL(pid_nr);
+
```

```
 struct pid *alloc_pid(void)
 {
  struct pid *pid;
  enum pid_type type;
- int nr = -1;
+ struct pid_nr *pid_nr;

  pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
  if (!pid)
-  goto out;
-
- nr = alloc_pidmap(task_pid_ns(current));
- if (nr < 0)
-  goto out_free;
+  return NULL;

  atomic_set(&pid->count, 1);
- pid->nr = nr;
+
  for (type = 0; type < PIDTYPE_MAX; ++type)
   INIT_HLIST_HEAD(&pid->tasks[type]);

- spin_lock_irq(&pidmap_lock);
- hlist_add_head_rcu(&pid->pid_chain, &pid_hash[pid_hashfn(pid->nr)]);
- spin_unlock_irq(&pidmap_lock);
+ INIT_HLIST_HEAD(&pid->pid_nrs);
+
+ pid_nr = alloc_pid_nr(task_pid_ns(current), pid);
+ if (!pid_nr)
+  goto out_free_pid;
+
+ pid->nr = pid_nr->nr;   /* pid->nr to be removed soon */
+ hlist_add_head(&pid_nr->node, &pid->pid_nrs);

-out:
  return pid;

-out_free:
+out_free_pid:
  kmem_cache_free(pid_cachep, pid);
- pid = NULL;
- goto out;
+ return NULL;
 }

 struct pid * fastcall find_pid(int nr)
 {
  struct hlist_node *elem;
```

```
- struct pid *pid;
+ struct pid_nr *pid_nr;
+ struct pid_namespace *ns = task_pid_ns(current);

- hlist_for_each_entry_rcu(pid, elem,
+ hlist_for_each_entry_rcu(pid_nr, elem,
    &pid_hash[pid_hashfn(nr)], pid_chain) {
-  if (pid->nr == nr)
-   return pid;
+  if (pid_nr->nr == nr && pid_nr->pid_ns == ns)
+   return pid_nr->pid;
  }
  return NULL;
 }
Index: 2.6.21-rc3-mm2/include/linux/pid.h
===================================================================
--- 2.6.21-rc3-mm2.orig/include/linux/pid.h
+++ 2.6.21-rc3-mm2/include/linux/pid.h
@@ -120,14 +120,7 @@ extern struct pid_nr *alloc_pid_nr(struc
    struct pid *pid);
 extern struct pid *alloc_pid(void);
 extern void FASTCALL(free_pid(struct pid *pid));
-
-static inline pid_t pid_nr(struct pid *pid)
-{
- pid_t nr = 0;
- if (pid)
-  nr = pid->nr;
- return nr;
-}
+extern pid_t pid_nr(struct pid *pid);

 #define do_each_pid_task(pid, type, task)    \
  do {       \

--
```


Subject: Re: [RFC][PATCH 06/14] Populate pid_nrs list with entry for init-pid-ns
Posted by ebiederm on Wed, 21 Mar 2007 09:23:28 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com writes:

> From: Cedric Le Goater <clg@fr.ibm.com>
> Subject: [RFC][PATCH 06/14] Populate pid_nrs list with entry for init-pid-ns
>
> Create/destroy the pid->pid_nrs list - when allocating/freeing a struct pid.
> The pid_nrs list contains just a single struct pid_nr for now, (corresponding
> to init-pid-ns).
>
> To enable finding a process based on any of its pid_t values, replace the
> use struct pid_nr, rather than struct pid, in the pid_hash table.
>
> Finally, reimplement find_pid() and pid_nr() based on the pid_nrs list.
>
> Changelog:
> - [Serge Hallyn's comment]: Add comments on what pid->lock protects
> and that pid->nr will eventually go away.
> - [Eric Biederman and containers list comments]: Reworked patches
> to drop support for unsharing pid namespace and to replace
> struct pid with struct pid_nr in the pid_hash table.
>
> Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
> include/linux/pid.h | 9 ------
> kernel/pid.c | 73 +++++++++++++++++++++++++++++++++++-----------------
> 2 files changed, 50 insertions(+), 32 deletions(-)
>

> @@ -242,50 +247,70 @@ struct pid_nr *alloc_pid_nr(struct pid_n
> */
> pid_nr->pid = pid;
>
> + spin_lock_irq(&pidmap_lock);
> + hlist_add_head_rcu(&pid_nr->pid_chain, &pid_hash[pid_hashfn(nr)]);
> + spin_unlock_irq(&pidmap_lock);
> +

Hmm. I think we really need to hash in some of the bits of the
pid_namespace so processes with identical pids don't get put on
the same hash chain. It isn't critical but it is a good idea.

Eric

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

Subject: Re: [RFC][PATCH 06/14] Populate pid_nrs list with entry for init-pid-ns
Posted by ebiederm on Wed, 21 Mar 2007 09:38:14 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com writes:
> Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
> include/linux/pid.h | 9 ------
> kernel/pid.c | 73 +++------------------
> 2 files changed, 50 insertions(+), 32 deletions(-)
>

> @@ -242,50 +247,70 @@ struct pid_nr *alloc_pid_nr(struct pid_n
> */
> pid_nr->pid = pid;
>
> + spin_lock_irq(&pidmap_lock);
> + hlist_add_head_rcu(&pid_nr->pid_chain, &pid_hash[pid_hashfn(nr)]);
> + spin_unlock_irq(&pidmap_lock);
> +

Hmm. I think we really need to hash in some of the bits of the
pid_namespace so processes with identical pids don't get put on
the same hash chain. It isn't critical but it is a good idea.

Eric

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

Subject: Re: [RFC][PATCH 06/14] Populate pid_nrs list with entry for init-pid-ns
Posted by Herbert Poetzl on Fri, 23 Mar 2007 01:27:51 GMT
View Forum Message <> Reply to Message

On Wed, Mar 21, 2007 at 03:23:28AM -0600, Eric W. Biederman wrote:
> sukadev@us.ibm.com writes:
>
> > From: Cedric Le Goater <clg@fr.ibm.com>
> > Subject: [RFC][PATCH 06/14] Populate pid_nrs list with entry for init-pid-ns
> >
> > Create/destroy the pid->pid_nrs list - when allocating/freeing a struct pid.
> > The pid_nrs list contains just a single struct pid_nr for now, (corresponding
> > to init-pid-ns).
> >
> > To enable finding a process based on any of its pid_t values, replace the

> > use struct pid_nr, rather than struct pid, in the pid_hash table.
> >
> > Finally, reimplement find_pid() and pid_nr() based on the pid_nrs list.
> >
> > Changelog:
> > - [Serge Hallyn's comment]: Add comments on what pid->lock protects
> >   and that pid->nr will eventually go away.
> > - [Eric Biederman and containers list comments]: Reworked patches
> >   to drop support for unsharing pid namespace and to replace
> >   struct pid with struct pid_nr in the pid_hash table.
> >
> > Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> > Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> > ---
> > include/linux/pid.h | 9 ------
> > kernel/pid.c | 73 +++++++++++++++++++++++++++++++++++-----------------
> > 2 files changed, 50 insertions(+), 32 deletions(-)
> >
>
> > @@ -242,50 +247,70 @@ struct pid_nr *alloc_pid_nr(struct pid_n
> >   */
> >   pid_nr->pid = pid;
> >
> > + spin_lock_irq(&pidmap_lock);
> > + hlist_add_head_rcu(&pid_nr->pid_chain, &pid_hash[pid_hashfn(nr)]);
> > + spin_unlock_irq(&pidmap_lock);
> > +
>
> Hmm. I think we really need to hash in some of the bits of the
> pid_namespace so processes with identical pids don't get put on
> the same hash chain. It isn't critical but it is a good idea.

yes, can't hurt to distribute that a little, was
already thinking about something like this:

#define pid_hashfn(nr) hash_long((unsigned long)(nr ^ space), pidhash_shift)

as the hash_long should distribute any changes
quite evenly ...

best,
Herbert

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

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