Subject: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
Posted by Sukadev Bhattiprolu on Sat, 16 Jun 2007 23:01:27 GMT
View Forum Message <> Reply to Message

Subject: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

With multiple pid namespaces, a process would be known by several pid_t values, one in each pid namespace. To represent this, we introduce a 'struct upid' which associates a single pid_t value with a single pid namespace.

We then replace the pid->nr field in 'struct pid' with a list of struct upid' entries (referred to as 'pid->upid_list'). This list represents the multiple pid_t values of the process, one in each namespace. The current patch adds just one element to this list, corresponding to 'init_pid_ns'. Subsequent patches implement multiple pid namespaces and add more elements to the list.

The 'struct upid' also replaces 'struct pid' in the pid_hash table to enable us to find processes given a pid_t from any namespace (i.e we find 'struct upid' for a given pid_t and from the 'struct upid', we find the 'struct pid' of the process)

We finally reimplement find_pid() and pid_to_nr() to use pid->upid_list and remove unused fields from 'struct pid'.

Changelog:
 2.6.21-mm2-pidns3:

 - 'struct upid' used to be called 'struct pid_nr' and a list of these
    were hanging off of 'struct pid'. So, we renamed 'struct pid_nr'
    and now hold them in a statically sized array in 'struct pid' since
    the number of 'struct upid's for a process is known at process-
    creation time.

 2.6.21-rc3-mm2:

 - [Eric Biederman] Combine all logical changes into one patch
 - [Eric Biederman] Implement __pid_nr(pid_ns, pid) for use in procfs.
   (now called pid_to_nr_in_ns()).
 - [Serge Hallyn]: Remove (!pid_nr) check in free_pid_nr()

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
 fs/proc/array.c | 30 ++++++--

```
 fs/proc/base.c               |    9 ++
 include/linux/init_task.h    |   14 +++-
 include/linux/pid.h          |   62 +++++++++++++++---
 include/linux/pid_namespace.h |   15 ++++
 kernel/fork.c                |    2
 kernel/pid.c                 |  145 +++++++++++++++++++++++++++++++++---------
 7 files changed, 220 insertions(+), 57 deletions(-)

Index: lx26-22-rc4-mm2/include/linux/pid.h
===================================================================
--- lx26-22-rc4-mm2.orig/include/linux/pid.h 2007-06-15 18:44:50.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/pid.h 2007-06-15 19:47:58.000000000 -0700
@@ -16,6 +16,25 @@ enum pid_type
  PIDTYPE_MAX
 };

+struct pid_namespace;
+
+/*
+ * A struct upid holds a process identifier (or pid->nr) for a given
+ * pid namespace.
+ *
+ * A list of 'struct upid' entries is stored in the struct pid. This list
+ * is used to get the process identifier associated with the pid
+ * namespace it is being seen from.
+ */
+struct upid
+{
+ /* Try to keep pid_chain in the same cacheline as nr for find_pid */
+ struct hlist_node pid_chain;  /* link hash collisions on pid_hash */
+ int nr;     /* user space pid number */
+ struct pid_namespace *pid_ns;  /* pid namespace in which nr is valid */
+ struct pid *pid;   /* back to task's unique kernel pid */
+};
+
 /*
  * What is struct pid?
  *
@@ -48,12 +67,11 @@ enum pid_type
 struct pid
 {
  atomic_t count;
- /* Try to keep pid_chain in the same cacheline as nr for find_pid */
- int nr;
- struct hlist_node pid_chain;
  /* lists of tasks that use this pid */
  struct hlist_head tasks[PIDTYPE_MAX];
  struct rcu_head rcu;
```

```
+ int num_upids;
+ struct upid upid_list[1];
 };

 extern struct pid init_struct_pid;
@@ -100,16 +118,10 @@ extern struct pid *FASTCALL(find_pid(int
 extern struct pid *find_get_pid(int nr);
 extern struct pid *find_ge_pid(int nr);

-extern struct pid *alloc_pid(enum copy_process_type);
+extern struct pid *dup_struct_pid(enum copy_process_type);
 extern void FASTCALL(free_pid(struct pid *pid));

-static inline pid_t pid_to_nr(struct pid *pid)
-{
- pid_t nr = 0;
- if (pid)
-  nr = pid->nr;
- return nr;
-}
+extern pid_t pid_to_nr(struct pid *pid);

 #define do_each_pid_task(pid, type, task)    \
  do {        \
@@ -122,4 +134,32 @@ static inline pid_t pid_to_nr(struct pid
    }      \
  } while (0)

+/*
+ * Return the pid_t by which the process @pid is known in the pid
+ * namespace @ns.
+ *
+ * Return 0 if:
+ *  - @pid is NULL (eg: procfs calls this for task_pgrp(init_task)
+ *    which is NULL).
+ *
+ *  - process does not have pid_t in the namespace @ns (eg: parent
+ *    process of a child reaper does not exist in the child namespace.
+ *    A getppid() call by the child reaper results in 0).
+ */
+static inline pid_t pid_to_nr_in_ns(struct pid *pid, struct pid_namespace *ns)
+{
+ int i;
+ struct upid *upid;
+
+ if (!pid)
+  return 0;
+
```

```
+ for (i = 0; i < pid->num_upids; i++) {
+  upid = &pid->upid_list[i];
+  if (upid->pid_ns == ns)
+   return upid->nr;
+ }
+ return 0;
+}
+
 #endif /* _LINUX_PID_H */
Index: lx26-22-rc4-mm2/include/linux/init_task.h
===================================================================
--- lx26-22-rc4-mm2.orig/include/linux/init_task.h 2007-06-15 18:44:07.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/init_task.h 2007-06-15 19:47:58.000000000 -0700
@@ -89,19 +89,27 @@ extern struct nsproxy init_nsproxy;

 extern struct group_info init_groups;

-#define INIT_STRUCT_PID {      \
- .count   = ATOMIC_INIT(1),    \
- .nr  = 0,       \
+#define INIT_STRUCT_UPID {      \
+ .nr  = 0,      \
   /* Don't put this struct pid in pid_hash */   \
   .pid_chain = { .next = NULL, .pprev = NULL },  \
+ .pid_ns  = &init_pid_ns,     \
+ .pid  = &init_struct_pid,    \
+}
+
+#define INIT_STRUCT_PID {      \
+ .count   = ATOMIC_INIT(1),    \
   .tasks  = {      \
    { .first = &init_task.pids[PIDTYPE_PID].node },  \
    { .first = &init_task.pids[PIDTYPE_PGID].node }, \
    { .first = &init_task.pids[PIDTYPE_SID].node },  \
   },       \
   .rcu  = RCU_HEAD_INIT,    \
+ .num_upids = 1,      \
+ .upid_list  = { INIT_STRUCT_UPID },    \
 }

+
 #define INIT_PID_LINK(type)      \
 {       \
   .node = {      \
Index: lx26-22-rc4-mm2/kernel/pid.c
===================================================================
--- lx26-22-rc4-mm2.orig/kernel/pid.c 2007-06-15 18:44:50.000000000 -0700
+++ lx26-22-rc4-mm2/kernel/pid.c 2007-06-15 19:48:11.000000000 -0700
```

```
@@ -30,6 +30,7 @@
 #include <linux/init_task.h>

 #define pid_hashfn(nr) hash_long((unsigned long)nr, pidhash_shift)
+
 static struct hlist_head *pid_hash;
 static int pidhash_shift;
 static struct kmem_cache *pid_cachep;
@@ -179,10 +180,67 @@ static int next_pidmap(struct pid_namesp
  return -1;
 }

+static void clear_upid(struct upid *upid)
+{
+ /* We can be called with write_lock_irq(&tasklist_lock) held */
+ unsigned long flags;
+
+ free_pidmap(upid->pid_ns, upid->nr);
+
+ spin_lock_irqsave(&pidmap_lock, flags);
+ hlist_del_rcu(&upid->pid_chain);
+ spin_unlock_irqrestore(&pidmap_lock, flags);
+
+ put_pid_ns(upid->pid_ns);
+}
+
+static int init_upid(struct upid *upid, struct pid *pid,
+  struct pid_namespace *pid_ns)
+{
+ int nr;
+
+ nr = alloc_pidmap(pid_ns);
+ if (nr < 0)
+  return nr;
+
+ upid->pid_ns = get_pid_ns(pid_ns);
+ upid->nr = nr;
+
+ /*
+  * The pid and upid are created and destroyed at the same time,
+  * as a single unit.  We don't need to refcount the pid use
+  * here.
+  */
+ upid->pid = pid;
+
+ INIT_HLIST_NODE(&upid->pid_chain);
+ spin_lock_irq(&pidmap_lock);
+ hlist_add_head_rcu(&upid->pid_chain, &pid_hash[pid_hashfn(nr)]);
```

```
+ spin_unlock_irq(&pidmap_lock);
+
+ return 0;
+}
+
+/*
+ * Return the pid_t by which the process @pid is known in the active
+ * pid namespace of the caller.
+ *
+ * TODO: pid_to_nr() cannot be easily inlined due to dependencies between
+ *       'task_struct', 'nsproxy' and 'pid_namespace'. As a part of
+ *       implementing multiple pid namespaces, we remove 'pid_namespace'
+ *       from 'nsproxy' and inline this function.
+ */
+pid_t pid_to_nr(struct pid *pid)
+{
+ return pid_to_nr_in_ns(pid, task_active_pid_ns(current));
+}
+EXPORT_SYMBOL_GPL(pid_to_nr);
+
 fastcall void put_pid(struct pid *pid)
 {
  if (!pid)
   return;
+
  if ((atomic_read(&pid->count) == 1) ||
       atomic_dec_and_test(&pid->count))
   kmem_cache_free(pid_cachep, pid);
@@ -197,66 +255,91 @@ static void delayed_put_pid(struct rcu_h

 fastcall void free_pid(struct pid *pid)
 {
- /* We can be called with write_lock_irq(&tasklist_lock) held */
- unsigned long flags;
+ int i;
+ struct upid *upid;

  /* check this here to keep copy_process() cleaner */
  if (unlikely(pid == &init_struct_pid))
   return;

- spin_lock_irqsave(&pidmap_lock, flags);
- hlist_del_rcu(&pid->pid_chain);
- spin_unlock_irqrestore(&pidmap_lock, flags);
+ /* clear any upids that we actually initialized */
+ for (i = 0; i < pid->num_upids; i++) {
+  upid = &pid->upid_list[i];
+  if (!upid->pid_ns)
```

```
+   break;
+  clear_upid(upid);
+ }

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

-struct pid *alloc_pid(enum copy_process_type copy_src)
+static struct pid *alloc_struct_pid(int num_upids)
 {
  struct pid *pid;
  enum pid_type type;
- int nr = -1;
-
- /* check this here to keep copy_process() cleaner */
- if (unlikely(copy_src == COPY_IDLE_PROCESS))
-  return &init_struct_pid;

- pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
+ /* for now we only support one pid namespace */
+ BUG_ON(num_upids != 1);
+ pid = kmem_cache_zalloc(pid_cachep, GFP_KERNEL);
  if (!pid)
-  goto out;
-
- nr = alloc_pidmap(task_active_pid_ns(current));
- if (nr < 0)
-  goto out_free;
+  return NULL;

  atomic_set(&pid->count, 1);
- pid->nr = nr;
+ pid->num_upids = num_upids;
+
  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);
+ return pid;
+}
+
+struct pid *dup_struct_pid(enum copy_process_type copy_src)
+{
+ int rc;
+ int i;
```

```
+ int num_upids;
+ struct pid *pid;
+ struct upid *upid;
+ struct upid *parent_upid;
+ struct pid *parent_pid = task_pid(current);
+
+ /* check this here to keep copy_process() cleaner */
+ if (unlikely(copy_src == COPY_IDLE_PROCESS))
+  return &init_struct_pid;
+
+ num_upids = parent_pid->num_upids;
+
+ pid = alloc_struct_pid(num_upids);
+ if (!pid)
+  return NULL;
+
+ for (i = 0; i < num_upids; i++) {
+  upid = &pid->upid_list[i];
+  parent_upid = &parent_pid->upid_list[i];
+  rc = init_upid(upid, pid, parent_upid->pid_ns);
+  if (rc)
+   goto out_free_pid;
+ }

-out:
  return pid;

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

 struct pid * fastcall find_pid(int nr)
 {
  struct hlist_node *elem;
- struct pid *pid;
+ struct upid *upid;
+ struct pid_namespace *ns = task_active_pid_ns(current);
+
+ /* TODO: Include pid namspace in the hash function */

- hlist_for_each_entry_rcu(pid, elem,
+ hlist_for_each_entry_rcu(upid, elem,
    &pid_hash[pid_hashfn(nr)], pid_chain) {
```

```
-  if (pid->nr == nr)
-   return pid;
+  if ((upid->pid_ns == ns) && (upid->nr == nr))
+   return upid->pid;
  }
  return NULL;
 }
Index: lx26-22-rc4-mm2/fs/proc/array.c
===================================================================
--- lx26-22-rc4-mm2.orig/fs/proc/array.c 2007-06-15 18:44:44.000000000 -0700
+++ lx26-22-rc4-mm2/fs/proc/array.c 2007-06-15 18:44:50.000000000 -0700
@@ -75,6 +75,7 @@
 #include <linux/cpuset.h>
 #include <linux/rcupdate.h>
 #include <linux/delayacct.h>
+#include <linux/pid_namespace.h>

 #include <asm/uaccess.h>
 #include <asm/pgtable.h>
@@ -161,8 +162,18 @@ static inline char * task_state(struct t
  struct group_info *group_info;
  int g;
  struct fdtable *fdt = NULL;
+ pid_t ppid = 0;
+ pid_t tracer_pid = 0;
+ /* TODO get pid_ns from proc mnt rather than current */
+ struct pid_namespace *ns = task_active_pid_ns(current);

  rcu_read_lock();
+
+ if (pid_alive(p)) {
+  ppid = pid_to_nr_in_ns(task_parent_tgid(p), ns);
+  tracer_pid = pid_to_nr_in_ns(task_tracer_pid(p), ns);
+ }
+
  buffer += sprintf(buffer,
   "State:\t%s\n"
   "Tgid:\t%d\n"
@@ -172,9 +183,10 @@ static inline char * task_state(struct t
   "Uid:\t%d\t%d\t%d\t%d\n"
   "Gid:\t%d\t%d\t%d\t%d\n",
   get_task_state(p),
-  p->tgid, p->pid,
-  pid_alive(p) ? rcu_dereference(p->real_parent)->tgid : 0,
-  pid_alive(p) && p->ptrace ? rcu_dereference(p->parent)->pid : 0,
+  pid_to_nr_in_ns(task_tgid(p), ns),
+  pid_to_nr_in_ns(task_pid(p), ns),
+  ppid,
```

```
+  tracer_pid,
   p->uid, p->euid, p->suid, p->fsuid,
   p->gid, p->egid, p->sgid, p->fsgid);

@@ -362,6 +374,8 @@ static int do_task_stat(struct task_stru
  unsigned long rsslim = 0;
  char tcomm[sizeof(task->comm)];
  unsigned long flags;
+ /* TODO get pid_ns from proc mnt rather than current */
+ struct pid_namespace *ns = task_active_pid_ns(current);

  state = *get_task_state(task);
  vsize = eip = esp = 0;
@@ -384,7 +398,7 @@ static int do_task_stat(struct task_stru
   struct signal_struct *sig = task->signal;

   if (sig->tty) {
-   tty_pgrp = pid_to_nr(sig->tty->pgrp);
+   tty_pgrp = pid_to_nr_in_ns(sig->tty->pgrp, ns);
    tty_nr = new_encode_dev(tty_devnum(sig->tty));
   }

@@ -414,9 +428,9 @@ static int do_task_stat(struct task_stru
    stime += cputime_to_clock_t(sig->stime);
   }

-  sid = signal_session(sig);
-  pgid = process_group(task);
-  ppid = rcu_dereference(task->real_parent)->tgid;
+  sid = pid_to_nr_in_ns(task_session(task), ns);
+  pgid = pid_to_nr_in_ns(task_pgrp(task), ns);
+  ppid = pid_to_nr_in_ns(task_parent_tgid(task), ns);

   unlock_task_sighand(task, &flags);
  }
@@ -447,7 +461,7 @@ static int do_task_stat(struct task_stru
  res = sprintf(buffer,"%d (%s) %c %d %d %d %d %d %u %lu \
 %lu %lu %lu %lu %lu %ld %ld %ld %ld %d 0 %llu %lu %ld %lu %lu %lu %lu %lu \
 %lu %lu %lu %lu %lu %lu %lu %lu %d %d %u %u %llu\n",
-  task->pid,
+  pid_to_nr_in_ns(task_pid(task), ns),
   tcomm,
   state,
   ppid,
Index: lx26-22-rc4-mm2/fs/proc/base.c
===================================================================
--- lx26-22-rc4-mm2.orig/fs/proc/base.c 2007-06-15 18:44:07.000000000 -0700
+++ lx26-22-rc4-mm2/fs/proc/base.c 2007-06-15 19:26:40.000000000 -0700
```

```
@@ -73,6 +73,7 @@
 #include <linux/audit.h>
 #include <linux/poll.h>
 #include <linux/nsproxy.h>
+#include <linux/pid_namespace.h>
 #include <linux/oom.h>
 #include <linux/elf.h>
 #include "internal.h"
@@ -2295,13 +2296,15 @@ static struct task_struct *next_tgid(uns
 {
  struct task_struct *task;
  struct pid *pid;
+ /* TODO get pid_ns from proc mnt rather than current */
+ struct pid_namespace *ns = task_active_pid_ns(current);

  rcu_read_lock();
 retry:
  task = NULL;
  pid = find_ge_pid(tgid);
  if (pid) {
-  tgid = pid->nr + 1;
+  tgid = pid_to_nr_in_ns(pid, ns) + 1;
   task = pid_task(pid, PIDTYPE_PID);
   /* What we to know is if the pid we have find is the
    * pid of a thread_group_leader.  Testing for task
@@ -2341,6 +2344,8 @@ int proc_pid_readdir(struct file * filp,
  struct task_struct *reaper = get_proc_task(filp->f_path.dentry->d_inode);
  struct task_struct *task;
  int tgid;
+ /* TODO get pid_ns from proc mnt rather than current */
+ struct pid_namespace *ns = task_active_pid_ns(current);

  if (!reaper)
   goto out_no_task;
@@ -2355,7 +2360,7 @@ int proc_pid_readdir(struct file * filp,
  for (task = next_tgid(tgid);
       task;
       put_task_struct(task), task = next_tgid(tgid + 1)) {
-  tgid = task->pid;
+  tgid = pid_to_nr_in_ns(task_pid(task), ns);
   filp->f_pos = tgid + TGID_OFFSET;
   if (proc_pid_fill_cache(filp, dirent, filldir, task, tgid) < 0) {
    put_task_struct(task);
Index: lx26-22-rc4-mm2/include/linux/pid_namespace.h
===================================================================
--- lx26-22-rc4-mm2.orig/include/linux/pid_namespace.h 2007-06-15 18:44:44.000000000 -0700
+++ lx26-22-rc4-mm2/include/linux/pid_namespace.h 2007-06-15 19:47:58.000000000 -0700
@@ -15,6 +15,18 @@ struct pidmap {
```

```
 #define PIDMAP_ENTRIES         ((PID_MAX_LIMIT + 8*PAGE_SIZE - 1)/PAGE_SIZE/8)

+/*
+ * Some properties/terminology for pid namespaces:
+ *
+ * Processes currently exist only in (or belong only to) the init_pid_ns.
+ * When we introduce the ability to clone the pid namespace, a process
+ * would exist in several namespaces. Off the many namespaces that the
+ * process can exist, one namespace is special and we refer to it as the
+ * 'active pid namespace' of the process.
+ *
+ * For now, we have only one pid namespace - init_pid_ns which is the
+ * 'active pid namespace' for all processes in the system.
+ */
 struct pid_namespace {
  struct kref kref;
  struct pidmap pidmap[PIDMAP_ENTRIES];
@@ -24,9 +36,10 @@ struct pid_namespace {

 extern struct pid_namespace init_pid_ns;

-static inline void get_pid_ns(struct pid_namespace *ns)
+static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
 {
  kref_get(&ns->kref);
+ return ns;
 }

 extern struct pid_namespace *copy_pid_ns(int flags, struct pid_namespace *ns);
Index: lx26-22-rc4-mm2/kernel/fork.c
===================================================================
--- lx26-22-rc4-mm2.orig/kernel/fork.c 2007-06-15 18:44:50.000000000 -0700
+++ lx26-22-rc4-mm2/kernel/fork.c 2007-06-15 19:47:58.000000000 -0700
@@ -1027,7 +1027,7 @@ static struct task_struct *copy_process(
  if (p->binfmt && !try_module_get(p->binfmt->module))
   goto bad_fork_cleanup_put_domain;

- pid = alloc_pid(copy_src);
+ pid = dup_struct_pid(copy_src);
  if (!pid)
   goto bad_fork_put_binfmt_module;


--
_____
Containers mailing list
Containers@lists.linux-foundation.org
```

Subject: Re: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
Posted by Pavel Emelianov on Mon, 18 Jun 2007 09:08:13 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Subject: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
>
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>
>
> With multiple pid namespaces, a process would be known by several pid_t
> values, one in each pid namespace. To represent this, we introduce a
> 'struct upid' which associates a single pid_t value with a single pid
> namespace.
>
> We then replace the pid->nr field in 'struct pid' with a list of struct upid'
> entries (referred to as 'pid->upid_list'). This list represents the multiple
> pid_t values of the process, one in each namespace. The current patch adds
> just one element to this list, corresponding to 'init_pid_ns'. Subsequent
> patches implement multiple pid namespaces and add more elements to the list.
>
> The 'struct upid' also replaces 'struct pid' in the pid_hash table to enable us
> to find processes given a pid_t from any namespace (i.e we find 'struct upid'
> for a given pid_t and from the 'struct upid', we find the 'struct pid' of the
> process)
>
> We finally reimplement find_pid() and pid_to_nr() to use pid->upid_list
> and remove unused fields from 'struct pid'.
>
> Changelog:
> 2.6.21-mm2-pidns3:
>
> - 'struct upid' used to be called 'struct pid_nr' and a list of these
> were hanging off of 'struct pid'. So, we renamed 'struct pid_nr'
> and now hold them in a statically sized array in 'struct pid' since
> the number of 'struct upid's for a process is known at process-
> creation time.
>
> 2.6.21-rc3-mm2:
>
> - [Eric Biederman] Combine all logical changes into one patch
> - [Eric Biederman] Implement __pid_nr(pid_ns, pid) for use in procfs.
> (now called pid_to_nr_in_ns()).
> - [Serge Hallyn]: Remove (!pid_nr) check in free_pid_nr()
>

> Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
> fs/proc/array.c | 30 ++++++--
> fs/proc/base.c | 9 ++
> include/linux/init_task.h | 14 +++-
> include/linux/pid.h | 62 +++++++++++++++---
> include/linux/pid_namespace.h | 15 ++++
> kernel/fork.c | 2
> kernel/pid.c | 145 +++++++++++++++++++++++++++++++++---------
> 7 files changed, 220 insertions(+), 57 deletions(-)
>
> Index: lx26-22-rc4-mm2/include/linux/pid.h
> ===
> --- lx26-22-rc4-mm2.orig/include/linux/pid.h 2007-06-15 18:44:50.000000000 -0700
> +++ lx26-22-rc4-mm2/include/linux/pid.h 2007-06-15 19:47:58.000000000 -0700
> @@ -16,6 +16,25 @@ enum pid_type
>   PIDTYPE_MAX
> };
>
> +struct pid_namespace;
> +
> +/*
> + * A struct upid holds a process identifier (or pid->nr) for a given
> + * pid namespace.
> + *
> + * A list of 'struct upid' entries is stored in the struct pid. This list
> + * is used to get the process identifier associated with the pid
> + * namespace it is being seen from.
> + */
> +struct upid
> +{
> + /* Try to keep pid_chain in the same cacheline as nr for find_pid */
> + struct hlist_node pid_chain; /* link hash collisions on pid_hash */
> + int nr;   /* user space pid number */
> + struct pid_namespace *pid_ns; /* pid namespace in which nr is valid */
> + struct pid *pid;   /* back to task's unique kernel pid */
> +};
> +
> /*
>   * What is struct pid?
>   *
> @@ -48,12 +67,11 @@ enum pid_type
> struct pid
> {
>   atomic_t count;
> - /* Try to keep pid_chain in the same cacheline as nr for find_pid */
> - int nr;

> - struct hlist_node pid_chain;
>   /* lists of tasks that use this pid */
>   struct hlist_head tasks[PIDTYPE_MAX];
>   struct rcu_head rcu;
> + int num_upids;
> + struct upid upid_list[1];

Further in your patches you define MAX_NESTED_PID_NS. What for, you
use the linked list here!?

> };
>
> extern struct pid init_struct_pid;

[snip]

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

---

Subject: Re: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
Posted by Sukadev Bhattiprolu on Mon, 18 Jun 2007 17:06:13 GMT
View Forum Message <> Reply to Message

Pavel Emelianov [xemul@openvz.org] wrote:
| sukadev@us.ibm.com wrote:
| > Subject: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
| >
| > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| >
| >
| > With multiple pid namespaces, a process would be known by several pid_t
| > values, one in each pid namespace. To represent this, we introduce a
| > 'struct upid' which associates a single pid_t value with a single pid
| > namespace.
| >
| > We then replace the pid->nr field in 'struct pid' with a list of struct upid'
| > entries (referred to as 'pid->upid_list'). This list represents the multiple
| > pid_t values of the process, one in each namespace. The current patch adds
| > just one element to this list, corresponding to 'init_pid_ns'. Subsequent
| > patches implement multiple pid namespaces and add more elements to the list.
| >
| > The 'struct upid' also replaces 'struct pid' in the pid_hash table to enable us
| > to find processes given a pid_t from any namespace (i.e we find 'struct upid'
| > for a given pid_t and from the 'struct upid', we find the 'struct pid' of the
| > process)
| >

| > We finally reimplement find_pid() and pid_to_nr() to use pid->upid_list
| > and remove unused fields from 'struct pid'.
| >
| > Changelog:
| > 2.6.21-mm2-pidns3:
| >
| > - 'struct upid' used to be called 'struct pid_nr' and a list of these
| >   were hanging off of 'struct pid'. So, we renamed 'struct pid_nr'
| >   and now hold them in a statically sized array in 'struct pid' since
| >   the number of 'struct upid's for a process is known at process-
| >   creation time.
| >
| > 2.6.21-rc3-mm2:
| >
| > - [Eric Biederman] Combine all logical changes into one patch
| > - [Eric Biederman] Implement __pid_nr(pid_ns, pid) for use in procfs.
| >   (now called pid_to_nr_in_ns()).
| > - [Serge Hallyn]: Remove (!pid_nr) check in free_pid_nr()
| >
| > Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
| > Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| > ---
| > fs/proc/array.c | 30 ++++++--
| > fs/proc/base.c | 9 ++
| > include/linux/init_task.h | 14 +++-
| > include/linux/pid.h | 62 +++++++++++++---
| > include/linux/pid_namespace.h | 15 ++++
| > kernel/fork.c | 2
| > kernel/pid.c | 145 +++++++++++++++++++++++++++++++++++++---------
| > 7 files changed, 220 insertions(+), 57 deletions(-)
| >
| > Index: lx26-22-rc4-mm2/include/linux/pid.h
| > ===
| > --- lx26-22-rc4-mm2.orig/include/linux/pid.h 2007-06-15 18:44:50.000000000 -0700
| > +++ lx26-22-rc4-mm2/include/linux/pid.h 2007-06-15 19:47:58.000000000 -0700
| > @@ -16,6 +16,25 @@ enum pid_type
| >   PIDTYPE_MAX
| > };
| >
| > +struct pid_namespace;
| > +
| > +/*
| > + * A struct upid holds a process identifier (or pid->nr) for a given
| > + * pid namespace.
| > + *
| > + * A list of 'struct upid' entries is stored in the struct pid. This list
| > + * is used to get the process identifier associated with the pid
| > + * namespace it is being seen from.

| > + */
| > +struct upid
| > +{
| > + /* Try to keep pid_chain in the same cacheline as nr for find_pid */
| > + struct hlist_node pid_chain; /* link hash collisions on pid_hash */
| > + int nr;   /* user space pid number */
| > + struct pid_namespace *pid_ns; /* pid namespace in which nr is valid */
| > + struct pid *pid;   /* back to task's unique kernel pid */
| > +};
| > +
| > /*
| >   * What is struct pid?
| >   *
| > @@ -48,12 +67,11 @@ enum pid_type
| > struct pid
| > {
| >   atomic_t count;
| > - /* Try to keep pid_chain in the same cacheline as nr for find_pid */
| > - int nr;
| > - struct hlist_node pid_chain;
| >   /* lists of tasks that use this pid */
| >   struct hlist_head tasks[PIDTYPE_MAX];
| >   struct rcu_head rcu;
| > + int num_upids;
| > + struct upid upid_list[1];
|
| Further in your patches you define MAX_NESTED_PID_NS. What for, you
| use the linked list here!?

Hmm. I don't understand. upid_list[] is an array (and not a linked
list). Are you saying the '_list' in 'upid_list' is misleading ?

Placing a limit like MAX_NESTED_PID_NS simplifies allocation of
'struct pid'.

|
| > };
| >
| > extern struct pid init_struct_pid;
|
| [snip]

Subject: Re: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
Posted by Sukadev Bhattiprolu on Tue, 19 Jun 2007 06:48:51 GMT
View Forum Message <> Reply to Message

Pavel Emelianov [xemul@openvz.org] wrote:
| sukadev@us.ibm.com wrote:
| > Pavel Emelianov [xemul@openvz.org] wrote:
| > | sukadev@us.ibm.com wrote:
| > | > Subject: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
| > | >
| > | > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| > | >
| > | >
| > | > With multiple pid namespaces, a process would be known by several pid_t
| > | > values, one in each pid namespace. To represent this, we introduce a
| > | > 'struct upid' which associates a single pid_t value with a single pid
| > | > namespace.
| > | >
| > | > We then replace the pid->nr field in 'struct pid' with a list of struct upid'
| > | > entries (referred to as 'pid->upid_list'). This list represents the multiple
| > | > pid_t values of the process, one in each namespace. The current patch adds
| > | > just one element to this list, corresponding to 'init_pid_ns'. Subsequent
| > | > patches implement multiple pid namespaces and add more elements to the list.
| > | >
| > | > The 'struct upid' also replaces 'struct pid' in the pid_hash table to enable us
| > | > to find processes given a pid_t from any namespace (i.e we find 'struct upid'
| > | > for a given pid_t and from the 'struct upid', we find the 'struct pid' of the
| > | > process)
| > | >
| > | > We finally reimplement find_pid() and pid_to_nr() to use pid->upid_list
| > | > and remove unused fields from 'struct pid'.
| > | >
| > | > Changelog:
| > | > 2.6.21-mm2-pidns3:
| > | >
| > | > - 'struct upid' used to be called 'struct pid_nr' and a list of these
| > | > were hanging off of 'struct pid'. So, we renamed 'struct pid_nr'
| > | > and now hold them in a statically sized array in 'struct pid' since
| > | > the number of 'struct upid's for a process is known at process-
| > | > creation time.
| > | >
| > | > 2.6.21-rc3-mm2:
| > | >
| > | > - [Eric Biederman] Combine all logical changes into one patch
| > | > - [Eric Biederman] Implement __pid_nr(pid_ns, pid) for use in procfs.
| > | > (now called pid_to_nr_in_ns()).
| > | > - [Serge Hallyn]: Remove (!pid_nr) check in free_pid_nr()
| > | >
| > | > Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

```
| > | > Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| > | > ---
| > | >  fs/proc/array.c              |   30 ++++++--
| > | >  fs/proc/base.c               |    9 ++
| > | >  include/linux/init_task.h    |   14 +++-
| > | >  include/linux/pid.h           |   62 +++++++++++++++---
| > | >  include/linux/pid_namespace.h |   15 ++++
| > | >  kernel/fork.c                |    2
| > | >  kernel/pid.c                 |  145 +++++++++++++++++++++++++++++++++++++---------
| > | >  7 files changed, 220 insertions(+), 57 deletions(-)
| > | >
| > | > Index: lx26-22-rc4-mm2/include/linux/pid.h
| > | > ===================================================================
| > | > --- lx26-22-rc4-mm2.orig/include/linux/pid.h 2007-06-15 18:44:50.000000000 -0700
| > | > +++ lx26-22-rc4-mm2/include/linux/pid.h 2007-06-15 19:47:58.000000000 -0700
| > | > @@ -16,6 +16,25 @@ enum pid_type
| > | >   PIDTYPE_MAX
| > | >  };
| > | >
| > | > +struct pid_namespace;
| > | > +
| > | > +/*
| > | > + * A struct upid holds a process identifier (or pid->nr) for a given
| > | > + * pid namespace.
| > | > + *
| > | > + * A list of 'struct upid' entries is stored in the struct pid. This list
| > | > + * is used to get the process identifier associated with the pid
| > | > + * namespace it is being seen from.
| > | > + */
| > | > +struct upid
| > | > +{
| > | > + /* Try to keep pid_chain in the same cacheline as nr for find_pid */
| > | > + struct hlist_node pid_chain;  /* link hash collisions on pid_hash */
| > | > + int nr;     /* user space pid number */
| > | > + struct pid_namespace *pid_ns;  /* pid namespace in which nr is valid */
| > | > + struct pid *pid;   /* back to task's unique kernel pid */
| > | > +};
| > | > +
| > | >  /*
| > | >   * What is struct pid?
| > | >   *
| > | > @@ -48,12 +67,11 @@ enum pid_type
| > | >  struct pid
| > | >  {
| > | >   atomic_t count;
| > | > - /* Try to keep pid_chain in the same cacheline as nr for find_pid */
| > | > - int nr;
| > | > - struct hlist_node pid_chain;
```

| > | >   /* lists of tasks that use this pid */
| > | >   struct hlist_head tasks[PIDTYPE_MAX];
| > | >   struct rcu_head rcu;
| > | > + int num_upids;
| > | > + struct upid upid_list[1];
| > |
| > | Further in your patches you define MAX_NESTED_PID_NS. What for, you
| > | use the linked list here!?
| >
| > Hmm. I don't understand. upid_list[] is an array (and not a linked
| > list). Are you saying the '_list' in 'upid_list' is misleading ?
|
| Oh, I see! You allocate all the upids in one chunk. I have missed
| that, sorry :)
|
| > Placing a limit like MAX_NESTED_PID_NS simplifies allocation of
| > 'struct pid'.
|
| How? If we have, say, 100-level namespace than we have to create
| the sizeof(struct pid) + 100 * sizeof(struct upid) bytes.

I should have been a little more clear.

I was comparing this with my previous version which did not have the
MAX_NESTED_PID_NS limit and allowed for arbitrary levels of nesting
(100 or even 1000 :-). Allocating that kind of 'struct pid' is more
complex and looks like an overkill at this time.

With a limit like MAX_NESTED_PID_NS, we could in theory create that
many pid caches, one for each level of nesting and use the appropriate
cache in clone().

|
| >
| > |
| > | > };
| > | >
| > | > extern struct pid init_struct_pid;
| > |
| > | [snip]
| > _____
| > Containers mailing list
| > Containers@lists.linux-foundation.org
| > https://lists.linux-foundation.org/mailman/listinfo/containers
| >
| > _____
| > Devel mailing list
| > Devel@openvz.org

| > https://openvz.org/mailman/listinfo/devel
| >

_____

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

---

Subject: Re: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
Posted by Pavel Emelianov on Tue, 19 Jun 2007 07:21:46 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Pavel Emelianov [xemul@openvz.org] wrote:
> | sukadev@us.ibm.com wrote:
> | > Subject: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
> | >
> | > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> | >
> | >
> | > With multiple pid namespaces, a process would be known by several pid_t
> | > values, one in each pid namespace. To represent this, we introduce a
> | > 'struct upid' which associates a single pid_t value with a single pid
> | > namespace.
> | >
> | > We then replace the pid->nr field in 'struct pid' with a list of struct upid'
> | > entries (referred to as 'pid->upid_list'). This list represents the multiple
> | > pid_t values of the process, one in each namespace. The current patch adds
> | > just one element to this list, corresponding to 'init_pid_ns'. Subsequent
> | > patches implement multiple pid namespaces and add more elements to the list.
> | >
> | > The 'struct upid' also replaces 'struct pid' in the pid_hash table to enable us
> | > to find processes given a pid_t from any namespace (i.e we find 'struct upid'
> | > for a given pid_t and from the 'struct upid', we find the 'struct pid' of the
> | > process)
> | >
> | > We finally reimplement find_pid() and pid_to_nr() to use pid->upid_list
> | > and remove unused fields from 'struct pid'.
> | >
> | > Changelog:
> | > 2.6.21-mm2-pidns3:
> | >
> | > - 'struct upid' used to be called 'struct pid_nr' and a list of these
> | > were hanging off of 'struct pid'. So, we renamed 'struct pid_nr'
> | > and now hold them in a statically sized array in 'struct pid' since
> | > the number of 'struct upid's for a process is known at process-
> | > creation time.
> | >

```
> | >  2.6.21-rc3-mm2:
> | >
> | >  - [Eric Biederman] Combine all logical changes into one patch
> | >  - [Eric Biederman] Implement __pid_nr(pid_ns, pid) for use in procfs.
> | >    (now called pid_to_nr_in_ns()).
> | >  - [Serge Hallyn]: Remove (!pid_nr) check in free_pid_nr()
> | >
> | > Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> | > Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> | > ---
> | >  fs/proc/array.c              |   30 ++++++--
> | >  fs/proc/base.c               |    9 ++
> | >  include/linux/init_task.h    |   14 +++-
> | >  include/linux/pid.h          |   62 +++++++++++++++---
> | >  include/linux/pid_namespace.h |   15 ++++
> | >  kernel/fork.c                |    2
> | >  kernel/pid.c                 |  145 +++++++++++++++++++++++++++++++++---------
> | >  7 files changed, 220 insertions(+), 57 deletions(-)
> | >
> | > Index: lx26-22-rc4-mm2/include/linux/pid.h
> | > ===================================================================
> | > --- lx26-22-rc4-mm2.orig/include/linux/pid.h 2007-06-15 18:44:50.000000000 -0700
> | > +++ lx26-22-rc4-mm2/include/linux/pid.h 2007-06-15 19:47:58.000000000 -0700
> | > @@ -16,6 +16,25 @@ enum pid_type
> | >    PIDTYPE_MAX
> | >  };
> | >
> | > +struct pid_namespace;
> | > +
> | > +/*
> | > + * A struct upid holds a process identifier (or pid->nr) for a given
> | > + * pid namespace.
> | > + *
> | > + * A list of 'struct upid' entries is stored in the struct pid. This list
> | > + * is used to get the process identifier associated with the pid
> | > + * namespace it is being seen from.
> | > + */
> | > +struct upid
> | > +{
> | > + /* Try to keep pid_chain in the same cacheline as nr for find_pid */
> | > + struct hlist_node pid_chain;  /* link hash collisions on pid_hash */
> | > + int nr;     /* user space pid number */
> | > + struct pid_namespace *pid_ns;  /* pid namespace in which nr is valid */
> | > + struct pid *pid;   /* back to task's unique kernel pid */
> | > +};
> | > +
> | >  /*
> | >   * What is struct pid?
```

> | > *
> | > @@ -48,12 +67,11 @@ enum pid_type
> | > struct pid
> | > {
> | > atomic_t count;
> | > - /* Try to keep pid_chain in the same cacheline as nr for find_pid */
> | > - int nr;
> | > - struct hlist_node pid_chain;
> | > /* lists of tasks that use this pid */
> | > struct hlist_head tasks[PIDTYPE_MAX];
> | > struct rcu_head rcu;
> | > + int num_upids;
> | > + struct upid upid_list[1];
> |
> | Further in your patches you define MAX_NESTED_PID_NS. What for, you
> | use the linked list here!?
>
> Hmm. I don't understand. upid_list[] is an array (and not a linked
> list). Are you saying the '_list' in 'upid_list' is misleading ?

Oh, I see! You allocate all the upids in one chunk. I have missed
that, sorry :)

> Placing a limit like MAX_NESTED_PID_NS simplifies allocation of
> 'struct pid'.

How? If we have, say, 100-level namespace than we have to create
the sizeof(struct pid) + 100 * sizeof(struct upid) bytes.

>
> |
> | > };
> | >
> | > extern struct pid init_struct_pid;
> |
> | [snip]
> _____
> 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

Subject: Re: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
Posted by Pavel Emelianov on Tue, 19 Jun 2007 07:50:55 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Pavel Emelianov [xemul@openvz.org] wrote:
> | sukadev@us.ibm.com wrote:
> | > Pavel Emelianov [xemul@openvz.org] wrote:
> | > | sukadev@us.ibm.com wrote:
> | > | > Subject: [PATCH 08/17] Pid-NS(V3) Define/use pid->upid_list list.
> | > | >
> | > | > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> | > | >
> | > | >
> | > | > With multiple pid namespaces, a process would be known by several pid_t
> | > | > values, one in each pid namespace. To represent this, we introduce a
> | > | > 'struct upid' which associates a single pid_t value with a single pid
> | > | > namespace.
> | > | >
> | > | > We then replace the pid->nr field in 'struct pid' with a list of struct upid'
> | > | > entries (referred to as 'pid->upid_list'). This list represents the multiple
> | > | > pid_t values of the process, one in each namespace. The current patch adds
> | > | > just one element to this list, corresponding to 'init_pid_ns'. Subsequent
> | > | > patches implement multiple pid namespaces and add more elements to the list.
> | > | >
> | > | > The 'struct upid' also replaces 'struct pid' in the pid_hash table to enable us
> | > | > to find processes given a pid_t from any namespace (i.e we find 'struct upid'
> | > | > for a given pid_t and from the 'struct upid', we find the 'struct pid' of the
> | > | > process)
> | > | >
> | > | > We finally reimplement find_pid() and pid_to_nr() to use pid->upid_list
> | > | > and remove unused fields from 'struct pid'.
> | > | >
> | > | > Changelog:
> | > | > 2.6.21-mm2-pidns3:
> | > | >
> | > | > - 'struct upid' used to be called 'struct pid_nr' and a list of these
> | > | > were hanging off of 'struct pid'. So, we renamed 'struct pid_nr'
> | > | > and now hold them in a statically sized array in 'struct pid' since
> | > | > the number of 'struct upid's for a process is known at process-
> | > | > creation time.
> | > | >
> | > | > 2.6.21-rc3-mm2:
> | > | >
> | > | > - [Eric Biederman] Combine all logical changes into one patch
> | > | > - [Eric Biederman] Implement __pid_nr(pid_ns, pid) for use in procfs.
> | > | > (now called pid_to_nr_in_ns()).
> | > | > - [Serge Hallyn]: Remove (!pid_nr) check in free_pid_nr()
> | > | >

> | > | > Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> | > | > Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> | > | > ---
> | > | > fs/proc/array.c | 30 ++++++--
> | > | > fs/proc/base.c | 9 ++
> | > | > include/linux/init_task.h | 14 +++-
> | > | > include/linux/pid.h | 62 +++++++++++++++---
> | > | > include/linux/pid_namespace.h | 15 ++++
> | > | > kernel/fork.c | 2
> | > | > kernel/pid.c | 145 +++++++++++++++++++++++++++++++++---------
> | > | > 7 files changed, 220 insertions(+), 57 deletions(-)
> | > | >
> | > | > Index: lx26-22-rc4-mm2/include/linux/pid.h
> | > | > ===
> | > | > --- lx26-22-rc4-mm2.orig/include/linux/pid.h 2007-06-15 18:44:50.000000000 -0700
> | > | > +++ lx26-22-rc4-mm2/include/linux/pid.h 2007-06-15 19:47:58.000000000 -0700
> | > | > @@ -16,6 +16,25 @@ enum pid_type
> | > | >   PIDTYPE_MAX
> | > | > };
> | > | >
> | > | > +struct pid_namespace;
> | > | > +
> | > | > +/*
> | > | > + * A struct upid holds a process identifier (or pid->nr) for a given
> | > | > + * pid namespace.
> | > | > + *
> | > | > + * A list of 'struct upid' entries is stored in the struct pid. This list
> | > | > + * is used to get the process identifier associated with the pid
> | > | > + * namespace it is being seen from.
> | > | > + */
> | > | > +struct upid
> | > | > +{
> | > | > + /* Try to keep pid_chain in the same cacheline as nr for find_pid */
> | > | > + struct hlist_node pid_chain; /* link hash collisions on pid_hash */
> | > | > + int nr;   /* user space pid number */
> | > | > + struct pid_namespace *pid_ns; /* pid namespace in which nr is valid */
> | > | > + struct pid *pid;   /* back to task's unique kernel pid */
> | > | > +};
> | > | > +
> | > | > /*
> | > | >   * What is struct pid?
> | > | >   *
> | > | > @@ -48,12 +67,11 @@ enum pid_type
> | > | > struct pid
> | > | > {
> | > | >   atomic_t count;
> | > | > - /* Try to keep pid_chain in the same cacheline as nr for find_pid */
> | > | > - int nr;

> | > | > - struct hlist_node pid_chain;
> | > | >   /* lists of tasks that use this pid */
> | > | >   struct hlist_head tasks[PIDTYPE_MAX];
> | > | >   struct rcu_head rcu;
> | > | > + int num_upids;
> | > | > + struct upid upid_list[1];
> | > |
> | > | Further in your patches you define MAX_NESTED_PID_NS. What for, you
> | > | use the linked list here!?
> | >
> | > Hmm. I don't understand. upid_list[] is an array (and not a linked
> | > list). Are you saying the '_list' in 'upid_list' is misleading ?
> |
> | Oh, I see! You allocate all the upids in one chunk. I have missed
> | that, sorry :)
> |
> | > Placing a limit like MAX_NESTED_PID_NS simplifies allocation of
> | > 'struct pid'.
> |
> | How? If we have, say, 100-level namespace than we have to create
> | the sizeof(struct pid) + 100 * sizeof(struct upid) bytes.
>
> I should have been a little more clear.
>
> I was comparing this with my previous version which did not have the
> MAX_NESTED_PID_NS limit and allowed for arbitrary levels of nesting
> (100 or even 1000 :-). Allocating that kind of 'struct pid' is more
> complex and looks like an overkill at this time.
>
> With a limit like MAX_NESTED_PID_NS, we could in theory create that
> many pid caches, one for each level of nesting and use the appropriate
> cache in clone().

Oh! I see. Thanks. Although this looks a bit ... weird.

> |
> | >
> | > |
> | > | > };
> | > | >
> | > | > extern struct pid init_struct_pid;
> | > |
> | > | [snip]
> | > _____
> | > Containers mailing list
> | > Containers@lists.linux-foundation.org
> | > https://lists.linux-foundation.org/mailman/listinfo/containers
> | >

> | > _____
> | > Devel mailing list
> | > Devel@openvz.org
> | > https://openvz.org/mailman/listinfo/devel
> | >
>

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