Subject: [PATCH 26/28] [MULTI 5/6] Multilevel model pid manipulations
Posted by Pavel Emelianov on Fri, 15 Jun 2007 16:28:27 GMT
View Forum Message <> Reply to Message

The alloc_pid(), free_pid() and put_pid() implementation for multilevel
model (see [PREP 10/14]). This model allocates the appropriate number
and hashed them into two tables.

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

---

 pid.c | 158
++
 1 files changed, 158 insertions(+)

--- ./kernel/pid.c.multicore 2007-06-15 15:38:59.000000000 +0400
+++ ./kernel/pid.c 2007-06-15 15:41:30.000000000 +0400
@@ -73,6 +73,22 @@ static struct hlist_head *pid_hash2;
 #define vpid_hash pid_hash2
 #define vpid_hashfn pid_ehashfn
 #endif
+
+#ifdef CONFIG_PID_NS_MULTILEVEL
+/*
+ * multilevel model stores pid numbers in two hashes
+ * - one hashes them with numerical id and the namespace
+ * - the other one - with struct pid and the namespace
+ */
+#define pid_phash pid_hash
+#define pid_nhash pid_hash2
+
+#define pid_phashfn pid_ehashfn
+#define pid_nhashfn pid_ehashfn
+
+struct kmem_cache *pid_nr_cachep;
+#endif
+
 #endif

 /*
@@ -377,6 +393,145 @@ static inline void del_pid_from_ns(struc
  pid->ns = &init_pid_ns;
 }
 #endif
+
+#ifdef CONFIG_PID_NS_MULTILEVEL
+static inline int alloc_pid_nr(struct pid *pid, struct pid_namespace *ns)

```
+{
+ struct pid_number *pnr;
+ int nr;
+
+ pnr = kmem_cache_alloc(pid_nr_cachep, GFP_KERNEL);
+ if (pnr == NULL)
+  goto out_nr;
+
+ nr = alloc_pidmap(ns);
+ if (nr < 0)
+  goto out_map;
+
+ get_pid_ns(ns);
+ pnr->nr = nr;
+ pnr->ns = ns;
+ pnr->pid = pid;
+ pnr->next = pid->pid_nrs;
+ pid->pid_nrs = pnr;
+ return 0;
+
+out_map:
+ kmem_cache_free(pid_nr_cachep, pnr);
+out_nr:
+ return -ENOMEM;
+}
+
+static inline void free_pid_nr(struct pid_number *pnr)
+{
+ free_pidmap(pnr->ns, pnr->nr);
+ put_pid_ns(pnr->ns);
+ kmem_cache_free(pid_nr_cachep, pnr);
+}
+
+static inline void hash_pid_nr(struct pid_number *pnr)
+{
+ hlist_add_head_rcu(&pnr->nr_chain,
+   &pid_nhash[pid_nhashfn(pnr->nr, pnr->ns)]);
+ hlist_add_head_rcu(&pnr->pid_chain,
+   &pid_phash[pid_phashfn(pnr->pid, pnr->ns)]);
+}
+
+static inline void unhash_pid_nr(struct pid_number *pnr)
+{
+ hlist_del_rcu(&pnr->nr_chain);
+ hlist_del_rcu(&pnr->pid_chain);
+}
+
+static inline void free_pid_nrs(struct pid *pid)
```

```
+{
+ struct pid_number *pnr;
+
+ while (pid->pid_nrs != NULL) {
+  pnr = pid->pid_nrs;
+  pid->pid_nrs = pnr->next;
+  free_pid_nr(pnr);
+ }
+}
+
+static inline int alloc_pid_nrs(struct pid *pid)
+{
+ int err;
+ struct pid *parent_pid;
+ struct pid_number *pnr;
+
+ pid->pid_nrs = NULL;
+
+ parent_pid = task_pid(current);
+ for_each_pid_nr(pnr, parent_pid) {
+  err = alloc_pid_nr(pid, pnr->ns);
+  if (err < 0)
+   goto out;
+ }
+
+ spin_lock_irq(&pidmap_lock);
+ for_each_pid_nr(pnr, pid)
+  hash_pid_nr(pnr);
+ spin_unlock_irq(&pidmap_lock);
+ return 0;
+
+out:
+ free_pid_nrs(pid);
+ return err;
+}
+
+static inline void unhash_pid_nrs(struct pid *pid)
+{
+ unsigned long flags;
+ struct pid_number *pnr;
+
+ spin_lock_irqsave(&pidmap_lock, flags);
+ for_each_pid_nr(pnr, pid)
+  unhash_pid_nr(pnr);
+ spin_unlock_irqrestore(&pidmap_lock, flags);
+}
+
+struct pid_number *find_nr_by_pid(struct pid *pid, struct pid_namespace *ns)
```

```
+{
+ struct pid_number *pnr;
+ struct hlist_node *n;
+
+ hlist_for_each_entry_rcu(pnr, n, &pid_phash[pid_phashfn(pid, ns)],
+   pid_chain)
+  if (pnr->pid == pid && pnr->ns == ns)
+   return pnr;
+
+ return NULL;
+}
+
+EXPORT_SYMBOL_GPL(find_nr_by_pid);
+
+/*
+ * this function finds the struct pid_number by its numerical id
+ * the name is not so beautiful, but this is an internal function
+ */
+
+static struct pid_number *find_nr_by_nr(pid_t nr, struct pid_namespace *ns)
+{
+ struct pid_number *pnr;
+ struct hlist_node *n;
+
+ hlist_for_each_entry_rcu(pnr, n, &pid_nhash[pid_nhashfn(nr, ns)],
+   nr_chain)
+  if (pnr->nr == nr && pnr->ns == ns)
+   return pnr;
+
+ return NULL;
+}
+
+struct pid * fastcall find_pid_ns(int nr, struct pid_namespace *ns)
+{
+ struct pid_number *pnr;
+
+ pnr = find_nr_by_nr(nr, ns);
+ return pnr != NULL ? pnr->pid : NULL;
+}
+#endif
 #endif

 EXPORT_SYMBOL_GPL(find_pid_ns);
@@ -762,4 +917,7 @@ void __init pidmap_init(void)
  atomic_dec(&init_pid_ns.pidmap[0].nr_free);

  pid_cachep = KMEM_CACHE(pid, SLAB_PANIC);
+#ifdef CONFIG_PID_NS_MULTILEVEL
```

+ pid_nr_cachep = KMEM_CACHE(pid_number, SLAB_PANIC);
+#endif
 }

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