Subject: [PATCH 20/28] [FLAT 5/6] Flat model pid manipulations
Posted by Pavel Emelianov on Fri, 15 Jun 2007 16:20:19 GMT
View Forum Message <> Reply to Message

The alloc_pid(), free_pid() and put_pid() implementation for flat
model (see [PREP 10/14]). This model allocates ids from two maps and
hashes the pid in two tables.

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

```
 pid.c |   99
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 files changed, 99 insertions(+)

--- ./kernel/pid.c.flatcore 2007-06-15 15:14:33.000000000 +0400
+++ ./kernel/pid.c 2007-06-15 15:26:23.000000000 +0400
@@ -64,6 +64,12 @@ static inline int mk_pid(struct pid_name
 static struct hlist_head *pid_hash2;
 #define pid_ehashfn(nr, ns) hash_long((unsigned long)nr + (unsigned long)ns, \
     pidhash_shift)
+
+#ifdef CONFIG_PID_NS_FLAT
+/* flat model uses the second hash to find the pids by their virtual nrs */
+#define vpid_hash pid_hash2
+#define vpid_hashfn pid_ehashfn
+#endif
 #endif

 /*
@@ -233,6 +239,99 @@ struct pid * fastcall find_pid_ns(int nr
  return NULL;
 }
 #else
+#ifdef CONFIG_PID_NS_FLAT
+static inline int alloc_pid_nrs(struct pid *pid)
+{
+ int nr, vnr;
+ struct pid_namespace *ns;
+
+ vnr = nr = alloc_pidmap(&init_pid_ns);
+ if (nr < 0)
+  goto out;
+
+ ns = current->nsproxy->pid_ns;
+ /*
+  * pids in init namespace have both nr and vnr equal
```

```
+  * pids in subnamespace hold the namespace
+  */
+ if (ns != &init_pid_ns) {
+  vnr = alloc_pidmap(ns);
+  if (vnr < 0)
+   goto out_vnr;
+
+  get_pid_ns(ns);
+ }
+
+ pid->nr = nr;
+ pid->vnr = vnr;
+ pid->ns = ns;
+ spin_lock_irq(&pidmap_lock);
+ hlist_add_head_rcu(&pid->pid_chain, &pid_hash[pid_hashfn(nr)]);
+ if (ns != &init_pid_ns)
+  hlist_add_head_rcu(&pid->vpid_chain,
+    &vpid_hash[vpid_hashfn(vnr, ns)]);
+ spin_unlock_irq(&pidmap_lock);
+ return 0;
+
+out_vnr:
+ free_pidmap(&init_pid_ns, nr);
+out:
+ return vnr;
+}
+
+static inline void unhash_pid_nrs(struct pid *pid)
+{
+ unsigned long flags;
+
+ spin_lock_irqsave(&pidmap_lock, flags);
+ hlist_del_rcu(&pid->pid_chain);
+ if (pid->ns != &init_pid_ns)
+  hlist_del_rcu(&pid->vpid_chain);
+ spin_unlock_irqrestore(&pidmap_lock, flags);
+
+ free_pidmap(&init_pid_ns, pid->nr);
+ if (pid->ns != &init_pid_ns)
+  free_pidmap(pid->ns, pid->vnr);
+}
+
+static inline void free_pid_nrs(struct pid *pid)
+{
+ if (pid->ns != &init_pid_ns)
+  put_pid_ns(pid->ns);
+}
+
```

```
+static inline struct pid *find_global_pid(int nr)
+{
+ struct pid *pid;
+ struct hlist_node *elem;
+
+ hlist_for_each_entry_rcu(pid, elem,
+   &pid_hash[pid_hashfn(nr)], pid_chain) {
+  if (pid->nr == nr)
+   return pid;
+ }
+ return NULL;
+}
+
+static inline struct pid *find_virtual_pid(int nr, struct pid_namespace *ns)
+{
+ struct pid *pid;
+ struct hlist_node *elem;
+
+ hlist_for_each_entry_rcu(pid, elem,
+   &vpid_hash[vpid_hashfn(nr, ns)], vpid_chain) {
+  if (pid->vnr == nr && pid->ns == ns)
+   return pid;
+ }
+ return NULL;
+}
+
+struct pid * fastcall find_pid_ns(int nr, struct pid_namespace *ns)
+{
+ return (ns == &init_pid_ns ?
+   find_global_pid(nr) : find_virtual_pid(nr, ns));
+}
+#endif
 #endif

 EXPORT_SYMBOL_GPL(find_pid_ns);
```

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