Subject: [RFC][PATCH 1/5] Virtualization/containers: startup
Posted by Kirill Korotaev on Fri, 03 Feb 2006 16:57:18 GMT
View Forum Message <> Reply to Message

This patch introduces some abstract container/VPS kernel structure and
tiny amount of operations on it.

Patches following this one will be used for virtualization of some
resources based on this container infrastructure, including those VPIDs
patches I sent before.

What is important here is that:
- each VPS has unique ID
- each process in kernel can belong to one VPS only

Kirill

```
--- ./include/linux/vps_info.h.vps_info 2006-02-03 16:38:33.000000000 +0300
+++ ./include/linux/vps_info.h 2006-02-03 16:49:26.000000000 +0300
@@ -0,0 +1,41 @@
+#ifndef __LINUX_VPS_INFO_H_
+#define __LINUX_VPS_INFO_H_
+
+#include <asm/types.h>
+#include <asm/atomic.h>
+
+struct task_struct;
+
+struct vps_info {
+ u32 id;
+ struct task_struct *init_task;
+ atomic_t refcnt;
+};
+
+extern struct vps_info host_vps_info;
+typedef struct vps_info *vps_t;
+
+static inline vps_t get_vps(vps_t vps)
+{
+ atomic_inc(&vps->refcnt);
+ return vps;
+}
+
+static inline void put_vps(vps_t vps)
+{
+ atomic_dec(&vps->refcnt);
+}
+
```

```
+#include <linux/sched.h>
+#include <asm/current.h>
+
+static inline vps_t set_current_vps(vps_t vps)
+{
+ vps_t ret;
+
+ ret = current->owner_vps;
+ current->owner_vps = vps;
+ return ret;
+}
+
+#endif
--- ./include/linux/sched.h.vps_info 2006-02-03 16:38:33.000000000 +0300
+++ ./include/linux/sched.h 2006-02-03 16:43:42.000000000 +0300
@@ -687,6 +687,7 @@

 struct audit_context;  /* See audit.c */
 struct mempolicy;
+struct vps_info;

 struct task_struct {
  volatile long state; /* -1 unrunnable, 0 runnable, >0 stopped */
@@ -845,6 +846,7 @@
  struct backing_dev_info *backing_dev_info;

  struct io_context *io_context;
+ struct vps_info *owner_vps;

  unsigned long ptrace_message;
  siginfo_t *last_siginfo; /* For ptrace use.  */
@@ -875,6 +877,8 @@
  struct rcu_head rcu;
 };

+#define current_vps() (current->owner_vps)
+
 static inline pid_t process_group(struct task_struct *tsk)
 {
  return tsk->signal->pgrp;
--- ./include/linux/init_task.h.vps_info 2006-02-03 16:38:33.000000000 +0300
+++ ./include/linux/init_task.h 2006-02-03 16:43:18.000000000 +0300
@@ -3,6 +3,7 @@

 #include <linux/file.h>
 #include <linux/rcupdate.h>
+#include <linux/vps_info.h>
```

```
 #define INIT_FDTABLE \
 {       \
@@ -121,6 +122,7 @@
  .journal_info = NULL,      \
  .cpu_timers = INIT_CPU_TIMERS(tsk.cpu_timers),  \
  .fs_excl = ATOMIC_INIT(0),    \
+ .owner_vps = &host_vps_info,    \
 }


--- ./init/main.c.vps_info 2006-02-03 16:38:33.000000000 +0300
+++ ./init/main.c 2006-02-03 16:43:18.000000000 +0300
@@ -47,6 +47,7 @@
 #include <linux/rmap.h>
 #include <linux/mempolicy.h>
 #include <linux/key.h>
+#include <linux/vps_info.h>

 #include <asm/io.h>
 #include <asm/bugs.h>
@@ -434,6 +435,13 @@
  done = 1;
 }

+struct vps_info host_vps_info = {
+ .id  = 0,
+ .init_task = &init_task,
+ .refcnt  = ATOMIC_INIT(1),
+};
+
+EXPORT_SYMBOL(host_vps_info);
 /*
  * Activate the first processor.
  */
--- ./kernel/fork.c.vps_info 2006-02-03 16:38:33.000000000 +0300
+++ ./kernel/fork.c 2006-02-03 16:43:18.000000000 +0300
@@ -44,6 +44,7 @@
 #include <linux/rmap.h>
 #include <linux/acct.h>
 #include <linux/cn_proc.h>
+#include <linux/vps_info.h>

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -1132,6 +1133,7 @@
  p->ioprio = current->ioprio;

  SET_LINKS(p);
```

```
+ get_vps(p->owner_vps);
  if (unlikely(p->ptrace & PT_PTRACED))
   __ptrace_link(p, current->parent);

--- ./kernel/exit.c.vps_info 2006-02-03 16:38:33.000000000 +0300
+++ ./kernel/exit.c 2006-02-03 16:43:18.000000000 +0300
@@ -31,6 +31,7 @@
 #include <linux/signal.h>
 #include <linux/cn_proc.h>
 #include <linux/mutex.h>
+#include <linux/vps_info.h>

 #include <asm/uaccess.h>
 #include <asm/unistd.h>
@@ -107,6 +108,7 @@
  spin_unlock(&p->proc_lock);
  proc_pid_flush(proc_dentry);
  release_thread(p);
+ put_vps(p->owner_vps);
  put_task_struct(p);

  p = leader;
```