Subject: [PATCH 1/4] Virtualization/containers: introduction
Posted by Kirill Korotaev on Mon, 06 Feb 2006 21:55:44 GMT
View Forum Message <> Reply to Message

Hello,

I tried to take into account all the comments from you guys (thanks a lot for them!) and prepared a new version of virtualization patches. I will send only 4 patches today, just not to overflow everyone and keep it clear/tidy/possible to review.

This patch introduces some abstract container kernel structure and a number of operations on it.

The important properties of the proposed container implementation:
- each container has unique ID in the system
- each process in the kernel can belong to one container only
- effective container pointer (econtainer()) is used on the task to avoid insertion of additional argument "container" to all functions where it is required.
- kernel compilation with disabled virtualization should result in old good linux kernel

Patches following this one will be used for virtualization of the kernel resources based on this container infrastructure, including those VPID patches I sent before. Every virtualized resource can be given separate config option if needed (just give me to know if it is desired).

Signed-Off-By: Kirill Korotaev <dev@openvz.org>

Kirill

P.S. I understand that this virtualization spam can be unintersting for some of you, just give me to know if you want to be removed from CC.

P.P.S. All patches are against 2.6.16-rc2-git2 and compile when virtualization=n.

```
--- ./include/linux/container.h.vpsinfo 2006-02-06 22:35:36.000000000 +0300
+++ ./include/linux/container.h 2006-02-07 00:52:57.000000000 +0300
@@ -0,0 +1,59 @@
+#ifndef __LINUX_CONTAINER_H__
+#define __LINUX_CONTAINER_H__
+
+#include <linux/config.h>
+#include <asm/types.h>
+#include <asm/atomic.h>
```

```
+
+struct task_struct;
+
+struct container {
+ u32 id;
+ struct task_struct *init_task;
+ atomic_t refcnt;
+};
+
+extern struct container init_container;
+
+#ifdef CONFIG_CONTAINER
+
+static inline struct container *get_container(struct container *cont)
+{
+ atomic_inc(&cont->refcnt);
+ return cont;
+}
+
+static inline void put_container(struct container *cont)
+{
+ atomic_dec(&container->refcnt);
+}
+
+#include <linux/sched.h>
+#include <asm/current.h>
+
+static inline struct container *set_econtainer(struct container *cont)
+{
+ struct container *ret;
+
+ ret = current->econtainer;
+ current->econtainer = cont;
+ return ret;
+}
+
+#define econtainer()  (current->econtainer)
+
+extern void init_containers(void);
+
+#else /* CONFIG_CONTAINER */
+
+#define get_container(cont) (NULL)
+#define put_container(cont) do { } while (0)
+
+#define set_econtainer(cont) ((struct container *)NULL)
+#define econtainer()  (NULL)
+
```

```
+#define init_containers() do { } while (0)
+
+#endif /* CONFIG_CONTAINER */
+
+#endif
--- ./include/linux/init_task.h.vpsinfo 2006-01-03 06:21:10.000000000 +0300
+++ ./include/linux/init_task.h 2006-02-07 00:52:52.000000000 +0300
@@ -3,6 +3,7 @@

 #include <linux/file.h>
 #include <linux/rcupdate.h>
+#include <linux/container.h>

 #define INIT_FDTABLE \
 {       \
@@ -131,5 +132,7 @@ extern struct group_info init_groups;
  LIST_HEAD_INIT(cpu_timers[2]),     \
 }

+#define INIT_CONTAINER(cont)      \
+ .refcnt  = ATOMIC_INIT(1)

 #endif
--- ./include/linux/sched.h.vpsinfo 2006-02-06 22:15:05.000000000 +0300
+++ ./include/linux/sched.h 2006-02-07 00:53:32.000000000 +0300
@@ -687,6 +687,7 @@ static inline void prefetch_stack(struct

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

 struct task_struct {
  volatile long state; /* -1 unrunnable, 0 runnable, >0 stopped */
@@ -846,6 +847,11 @@ struct task_struct {

  struct io_context *io_context;

+#ifdef CONFIG_CONTAINER
+ struct container *container;
+ struct container *econtainer; /* effective container */
+#endif
+
  unsigned long ptrace_message;
  siginfo_t *last_siginfo; /* For ptrace use.  */
 /*
--- ./init/main.c.vpsinfo 2006-02-06 22:15:06.000000000 +0300
+++ ./init/main.c 2006-02-07 00:46:12.000000000 +0300
@@ -47,6 +47,7 @@
```

```
 #include <linux/rmap.h>
 #include <linux/mempolicy.h>
 #include <linux/key.h>
+#include <linux/container.h>

 #include <asm/io.h>
 #include <asm/bugs.h>
@@ -603,6 +604,8 @@ static void __init do_initcalls(void)
  */
 static void __init do_basic_setup(void)
 {
+ init_containers();
+
  /* drivers will send hotplug events */
  init_workqueues();
  usermodehelper_init();
--- ./kernel/Makefile.vpsinfo 2006-02-06 22:15:06.000000000 +0300
+++ ./kernel/Makefile 2006-02-07 00:46:12.000000000 +0300
@@ -34,6 +34,7 @@ obj-$(CONFIG_DETECT_SOFTLOCKUP) += softl
 obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
 obj-$(CONFIG_SECCOMP) += seccomp.o
 obj-$(CONFIG_RCU_TORTURE_TEST) += rcutorture.o
+obj-$(CONFIG_CONTAINER) += container.c

 ifneq ($(CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER),y)
 # According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
--- ./kernel/container.c.vpsinfo 2006-02-07 00:01:21.000000000 +0300
+++ ./kernel/container.c 2006-02-07 00:46:12.000000000 +0300
@@ -0,0 +1,16 @@
+#include <linux/container.h>
+
+/*
+ * Initial container.
+ *
+ * All tasks and other resources initially belong to it
+ */
+struct container init_container = INIT_CONTAINER(init_container);
+
+EXPORT_SYMBOL(init_container);
+
+void init_containers(void)
+{
+ /* remember who is init in container */
+ init_container.init_task = child_reaper;
+}
--- ./kernel/exit.c.vpsinfo 2006-02-06 22:15:06.000000000 +0300
+++ ./kernel/exit.c 2006-02-07 00:46:12.000000000 +0300
@@ -31,6 +31,7 @@
```

```
 #include <linux/signal.h>
 #include <linux/cn_proc.h>
 #include <linux/mutex.h>
+#include <linux/container.h>

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

  p = leader;
--- ./kernel/fork.c.vpsinfo 2006-02-06 22:15:06.000000000 +0300
+++ ./kernel/fork.c 2006-02-07 00:46:12.000000000 +0300
@@ -44,6 +44,7 @@
 #include <linux/rmap.h>
 #include <linux/acct.h>
 #include <linux/cn_proc.h>
+#include <linux/container.h>

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -1132,6 +1133,7 @@ static task_t *copy_process(unsigned lon
  p->ioprio = current->ioprio;

  SET_LINKS(p);
+ (void)get_container(p->container);
  if (unlikely(p->ptrace & PT_PTRACED))
   __ptrace_link(p, current->parent);
```