Subject: [RFC][PATCH 6/7] Account for the number of tasks within container
Posted by xemul on Tue, 06 Mar 2007 15:02:39 GMT
View Forum Message <> Reply to Message

Small and simple - each fork()/clone() is accounted
and rejected when limit is hit.

```
diff -upr linux-2.6.20.orig/include/linux/numproc_container.h
linux-2.6.20-0/include/linux/numproc_container.h
--- linux-2.6.20.orig/include/linux/numproc_container.h 2007-03-06 13:39:17.000000000 +0300
+++ linux-2.6.20-0/include/linux/numproc_container.h 2007-03-06 13:33:28.000000000 +0300
@@ -0,0 +1,32 @@
+#ifndef __NUMPROC_CONTAINER_H__
+#define __NUMPROC_CONTAINER_H__
+/*
+ * Numproc container
+ *
+ * Copyright 2007 OpenVZ SWsoft Inc
+ *
+ * Author: Pavel Emelianov <xemul@openvz.org>
+ *
+ */
+
+#ifdef CONFIG_PROCESS_CONTAINER
+int container_proc_charge(struct task_struct *tsk);
+void container_proc_uncharge(struct task_struct *tsk);
+
+void container_numproc_init_early(void);
+#else
+static inline int container_proc_charge(struct task_struct *tsk)
+{
+ return 0;
+}
+
+static inline void container_proc_uncharge(struct task_struct *tsk)
+{
+}
+
+static inline void container_numproc_init_early(void)
+{
+}
+#endif
+
+#endif
diff -upr linux-2.6.20.orig/include/linux/sched.h linux-2.6.20-0/include/linux/sched.h
--- linux-2.6.20.orig/include/linux/sched.h 2007-03-06 13:33:28.000000000 +0300
+++ linux-2.6.20-0/include/linux/sched.h 2007-03-06 13:33:28.000000000 +0300
@@ -1052,6 +1055,9 @@ struct task_struct {
```

```
 #ifdef CONFIG_FAULT_INJECTION
  int make_it_fail;
 #endif
+#ifdef CONFIG_PROCESS_CONTAINER
+ struct numproc_container *numproc_cnt;
+#endif
 };

 static inline pid_t process_group(struct task_struct *tsk)
diff -upr linux-2.6.20.orig/init/Kconfig linux-2.6.20-0/init/Kconfig
--- linux-2.6.20.orig/init/Kconfig 2007-03-06 13:33:28.000000000 +0300
+++ linux-2.6.20-0/init/Kconfig 2007-03-06 13:33:28.000000000 +0300
@@ -265,6 +265,12 @@ config CPUSETS
    Provides a simple Resource Controller for monitoring and
    controlling the total Resident Set Size of the tasks in a container

+config PROCESS_CONTAINER
+ bool "Numproc accounting container"
+ select RESOURCE_COUNTERS
+ help
+   Provides the-number-of-tasks accounting container
+
 config SYSFS_DEPRECATED
  bool "Create deprecated sysfs files"
  default y
diff -upr linux-2.6.20.orig/kernel/Makefile linux-2.6.20-0/kernel/Makefile
--- linux-2.6.20.orig/kernel/Makefile 2007-03-06 13:33:28.000000000 +0300
+++ linux-2.6.20-0/kernel/Makefile 2007-03-06 13:33:28.000000000 +0300
@@ -51,6 +51,7 @@ obj-$(CONFIG_RELAY) += relay.o
 obj-$(CONFIG_TASK_DELAY_ACCT) += delayacct.o
 obj-$(CONFIG_TASKSTATS) += taskstats.o tsacct.o
 obj-$(CONFIG_RESOURCE_COUNTERS) += res_counter.o
+obj-$(CONFIG_PROCESS_CONTAINER) += numproc_container.o

 ifneq ($(CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER),y)
 # According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
diff -upr linux-2.6.20.orig/kernel/exit.c linux-2.6.20-0/kernel/exit.c
--- linux-2.6.20.orig/kernel/exit.c 2007-03-06 13:33:28.000000000 +0300
+++ linux-2.6.20-0/kernel/exit.c 2007-03-06 13:33:28.000000000 +0300
@@ -48,6 +48,8 @@
 #include <asm/pgtable.h>
 #include <asm/mmu_context.h>

+#include <linux/numproc_container.h>
+
 extern void sem_exit (void);

 static void exit_mm(struct task_struct * tsk);
```

```
@@ -174,6 +176,7 @@ repeat:
  write_unlock_irq(&tasklist_lock);
  proc_flush_task(p);
  release_thread(p);
+ container_proc_uncharge(p);
  call_rcu(&p->rcu, delayed_put_task_struct);

  p = leader;
diff -upr linux-2.6.20.orig/kernel/fork.c linux-2.6.20-0/kernel/fork.c
--- linux-2.6.20.orig/kernel/fork.c 2007-03-06 13:33:28.000000000 +0300
+++ linux-2.6.20-0/kernel/fork.c 2007-03-06 13:33:28.000000000 +0300
@@ -57,6 +57,7 @@
 #include <asm/tlbflush.h>

 #include <linux/rss_container.h>
+#include <linux/numproc_container.h>

 /*
  * Protected counters by write_lock_irq(&tasklist_lock)
@@ -986,6 +999,9 @@ static struct task_struct *copy_process(
  if (!p)
   goto fork_out;

+ if (container_proc_charge(p))
+  goto charge_out;
+
  rt_mutex_init_task(p);

 #ifdef CONFIG_TRACE_IRQFLAGS
@@ -1302,6 +1318,8 @@ bad_fork_cleanup_count:
  atomic_dec(&p->user->processes);
  free_uid(p->user);
 bad_fork_free:
+ container_proc_uncharge(p);
+charge_out:
  free_task(p);
 fork_out:
  return ERR_PTR(retval);
diff -upr linux-2.6.20.orig/kernel/numproc_container.c linux-2.6.20-0/kernel/numproc_container.c
--- linux-2.6.20.orig/kernel/numproc_container.c 2007-03-06 13:39:17.000000000 +0300
+++ linux-2.6.20-0/kernel/numproc_container.c 2007-03-06 13:33:28.000000000 +0300
@@ -0,0 +1,151 @@
+/*
+ * Numproc accounting container
+ *
+ * Copyright 2007 OpenVZ SWsoft Inc
+ *
+ * Author: Pavel Emelianov <xemul@openvz.org>
```

```
+ *
+ */
+
+#include <linux/list.h>
+#include <linux/sched.h>
+#include <linux/mm.h>
+#include <linux/res_counter.h>
+#include <linux/numproc_container.h>
+
+static struct container_subsys numproc_subsys;
+
+struct numproc_container {
+ struct res_counter res;
+ struct container_subsys_state css;
+};
+
+static inline struct numproc_container *numproc_from_cont(struct container *cnt)
+{
+ return container_of(container_subsys_state(cnt, &numproc_subsys),
+   struct numproc_container, css);
+}
+
+int container_proc_charge(struct task_struct *new)
+{
+ struct numproc_container *np;
+
+ rcu_read_lock();
+ np = numproc_from_cont(task_container(current, &numproc_subsys));
+ css_get_current(&np->css);
+ rcu_read_unlock();
+
+ if (res_counter_charge(&np->res, 1)) {
+  css_put(&np->css);
+  return -ENOMEM;
+ }
+
+ new->numproc_cnt = np;
+ return 0;
+}
+
+void container_proc_uncharge(struct task_struct *tsk)
+{
+ struct numproc_container *np;
+
+ np = tsk->numproc_cnt;
+ res_counter_uncharge(&np->res, 1);
+ css_put(&np->css);
+}
```

```
+
+static int numproc_create(struct container_subsys *ss, struct container *cont)
+{
+ struct numproc_container *np;
+
+ np = kzalloc(sizeof(struct numproc_container), GFP_KERNEL);
+ if (np == NULL)
+  return -ENOMEM;
+
+ res_counter_init(&np->res);
+ cont->subsys[numproc_subsys.subsys_id] = &np->css;
+ return 0;
+}
+
+static void numproc_destroy(struct container_subsys *ss,
+  struct container *cont)
+{
+ kfree(numproc_from_cont(cont));
+}
+
+
+static ssize_t numproc_read(struct container *cont, struct cftype *cft,
+  struct file *file, char __user *userbuf,
+  size_t nbytes, loff_t *ppos)
+{
+ return res_counter_read(&numproc_from_cont(cont)->res, cft->private,
+   userbuf, nbytes, ppos);
+}
+
+static ssize_t numproc_write(struct container *cont, struct cftype *cft,
+  struct file *file, const char __user *userbuf,
+  size_t nbytes, loff_t *ppos)
+{
+ return res_counter_write(&numproc_from_cont(cont)->res, cft->private,
+   userbuf, nbytes, ppos);
+}
+
+
+static struct cftype numproc_usage = {
+ .name = "numproc_usage",
+ .private = RES_USAGE,
+ .read = numproc_read,
+};
+
+static struct cftype numproc_limit = {
+ .name = "numproc_limit",
+ .private = RES_LIMIT,
+ .read = numproc_read,
```

```
+ .write = numproc_write,
+};
+
+static struct cftype numproc_failcnt = {
+ .name = "numproc_failcnt",
+ .private = RES_FAILCNT,
+ .read = numproc_read,
+};
+
+static int numproc_populate(struct container_subsys *ss,
+  struct container *cont)
+{
+ int rc;
+
+ if ((rc = container_add_file(cont, &numproc_usage)) < 0)
+  return rc;
+ if ((rc = container_add_file(cont, &numproc_failcnt)) < 0)
+  return rc;
+ if ((rc = container_add_file(cont, &numproc_limit)) < 0)
+  return rc;
+
+ return 0;
+}
+
+static struct numproc_container init_numproc_container;
+
+static __init int numproc_create_early(struct container_subsys *ss,
+  struct container *cont)
+{
+ struct numproc_container *np;
+
+ np = &init_numproc_container;
+ res_counter_init(&np->res);
+ cont->subsys[numproc_subsys.subsys_id] = &np->css;
+ ss->create = numproc_create;
+ return 0;
+}
+
+static struct container_subsys numproc_subsys = {
+ .name = "numproc",
+ .create = numproc_create_early,
+ .destroy = numproc_destroy,
+ .populate = numproc_populate,
+};
+
+void __init container_numproc_init_early(void)
+{
+ container_register_subsys(&numproc_subsys);
```

```
+}
diff -upr linux-2.6.20.orig/kernel/container.c linux-2.6.20-0/kernel/container.c
--- linux-2.6.20.orig/kernel/container.c 2007-03-06 13:33:28.000000000 +0300
+++ linux-2.6.20-0/kernel/container.c 2007-03-06 13:35:48.000000000 +0300
@@ -60,6 +60,7 @@
 #include <linux/mutex.h>

 #include <linux/rss_container.h>
+#include <linux/numproc_container.h>

 #define CONTAINER_SUPER_MAGIC  0x27e0eb

@@ -1721,6 +1725,7 @@ int __init container_init_early(void)
  init_task.containers = &init_container_group;

  container_rss_init_early();
+ container_numproc_init_early();

  return 0;
 }
```