Subject: [PATCH 1/8] Resource counters
Posted by Pavel Emelianov on Mon, 04 Jun 2007 13:24:37 GMT
View Forum Message <> Reply to Message

Introduce generic structures and routines for resource accounting.

Each resource accounting container is supposed to aggregate it,
container_subsystem_state and its resource-specific members within.

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

---

diff -upr linux-2.6.22-rc2-mm1.orig/include/linux/res_counter.h
linux-2.6.22-rc2-mm1-2/include/linux/res_counter.h
--- linux-2.6.22-rc2-mm1.orig/include/linux/res_counter.h 2007-06-01 16:46:32.000000000 +0400
+++ linux-2.6.22-rc2-mm1-2/include/linux/res_counter.h 2007-06-01 16:35:15.000000000 +0400
@@ -0,0 +1,102 @@
+#ifndef __RES_COUNTER_H__
+#define __RES_COUNTER_H__
+
+/*
+ * resource counters
+ * contain common data types and routines for resource accounting
+ *
+ * Copyright 2007 OpenVZ SWsoft Inc
+ *
+ * Author: Pavel Emelianov <xemul@openvz.org>
+ *
+ */
+
+#include <linux/container.h>
+
+/*
+ * the core object. the container that wishes to account for some
+ * resource may include this counter into its structures and use
+ * the helpers described beyond
+ */
+
+struct res_counter {
+ /*
+ * the current resource consumption level
+ */
+ unsigned long usage;
+ /*
+ * the limit that usage cannot exceed
+ */
+ unsigned long limit;

```
+ /*
+  * the number of insuccessful attempts to consume the resource
+  */
+ unsigned long failcnt;
+ /*
+  * the lock to protect all of the above.
+  * the routines below consider this to be IRQ-safe
+  */
+ spinlock_t lock;
+};
+
+/*
+ * helpers to interact with userspace
+ * res_counter_read/_write - put/get the specified fields from the
+ * res_counter struct to/from the user
+ *
+ * @cnt:     the counter in question
+ * @member:  the field to work with (see RES_xxx below)
+ * @buf:     the buffer to opeate on,...
+ * @nbytes:  its size...
+ * @pos:     and the offset.
+ */
+
+ssize_t res_counter_read(struct res_counter *cnt, int member,
+  const char __user *buf, size_t nbytes, loff_t *pos);
+ssize_t res_counter_write(struct res_counter *cnt, int member,
+  const char __user *buf, size_t nbytes, loff_t *pos);
+
+/*
+ * the field descriptors. one for each member of res_counter
+ */
+
+enum {
+ RES_USAGE,
+ RES_LIMIT,
+ RES_FAILCNT,
+};
+
+/*
+ * helpers for accounting
+ */
+
+void res_counter_init(struct res_counter *cnt);
+
+/*
+ * charge - try to consume more resource.
+ *
+ * @cnt: the counter
```

```
+ * @val: the amount of the resource. each controller defines its own
+ *       units, e.g. numbers, bytes, Kbytes, etc
+ *
+ * returns 0 on success and <0 if the cnt->usage will exceed the cnt->limit
+ * _locked call expects the cnt->lock to be taken
+ */
+
+int res_counter_charge_locked(struct res_counter *cnt, unsigned long val);
+int res_counter_charge(struct res_counter *cnt, unsigned long val);
+
+/*
+ * uncharge - tell that some portion of the resource is released
+ *
+ * @cnt: the counter
+ * @val: the amount of the resource
+ *
+ * these calls check for usage underflow and show a warning on the console
+ * _locked call expects the cnt->lock to be taken
+ */
+
+void res_counter_uncharge_locked(struct res_counter *cnt, unsigned long val);
+void res_counter_uncharge(struct res_counter *cnt, unsigned long val);
+
+#endif
diff -upr linux-2.6.22-rc2-mm1.orig/init/Kconfig linux-2.6.22-rc2-mm1-2/init/Kconfig
--- linux-2.6.22-rc2-mm1.orig/init/Kconfig 2007-06-01 16:35:12.000000000 +0400
+++ linux-2.6.22-rc2-mm1-2/init/Kconfig 2007-06-01 16:35:13.000000000 +0400
@@ -328,6 +328,10 @@ config CPUSETS

    Say N if unsure.

+config RESOURCE_COUNTERS
+ bool
+ select CONTAINERS
+
 config SYSFS_DEPRECATED
  bool "Create deprecated sysfs files"
  default y
diff -upr linux-2.6.22-rc2-mm1.orig/kernel/Makefile linux-2.6.22-rc2-mm1-2/kernel/Makefile
--- linux-2.6.22-rc2-mm1.orig/kernel/Makefile 2007-06-01 16:35:12.000000000 +0400
+++ linux-2.6.22-rc2-mm1-2/kernel/Makefile 2007-06-01 16:35:13.000000000 +0400
@@ -56,6 +56,7 @@ obj-$(CONFIG_SYSCTL) += utsname_sysctl.o
 obj-$(CONFIG_UTS_NS) += utsname.o
 obj-$(CONFIG_TASK_DELAY_ACCT) += delayacct.o
 obj-$(CONFIG_TASKSTATS) += taskstats.o tsacct.o
+obj-$(CONFIG_RESOURCE_COUNTERS) += res_counter.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-2.6.22-rc2-mm1.orig/kernel/res_counter.c
linux-2.6.22-rc2-mm1-2/kernel/res_counter.c
--- linux-2.6.22-rc2-mm1.orig/kernel/res_counter.c 2007-06-01 16:46:32.000000000 +0400
+++ linux-2.6.22-rc2-mm1-2/kernel/res_counter.c 2007-06-01 16:35:15.000000000 +0400
@@ -0,0 +1,121 @@
+/*
+ * resource containers
+ *
+ * Copyright 2007 OpenVZ SWsoft Inc
+ *
+ * Author: Pavel Emelianov <xemul@openvz.org>
+ *
+ */
+
+#include <linux/types.h>
+#include <linux/parser.h>
+#include <linux/fs.h>
+#include <linux/res_counter.h>
+#include <linux/uaccess.h>
+
+void res_counter_init(struct res_counter *cnt)
+{
+ spin_lock_init(&cnt->lock);
+ cnt->limit = (unsigned long)LONG_MAX;
+}
+
+int res_counter_charge_locked(struct res_counter *cnt, unsigned long val)
+{
+ if (cnt->usage <= cnt->limit - val) {
+  cnt->usage += val;
+  return 0;
+ }
+
+ cnt->failcnt++;
+ return -ENOMEM;
+}
+
+int res_counter_charge(struct res_counter *cnt, unsigned long val)
+{
+ int ret;
+ unsigned long flags;
+
+ spin_lock_irqsave(&cnt->lock, flags);
+ ret = res_counter_charge_locked(cnt, val);
+ spin_unlock_irqrestore(&cnt->lock, flags);
+ return ret;
+}
```

```
+
+void res_counter_uncharge_locked(struct res_counter *cnt, unsigned long val)
+{
+ if (unlikely(cnt->usage < val)) {
+  WARN_ON(1);
+  val = cnt->usage;
+ }
+
+ cnt->usage -= val;
+}
+
+void res_counter_uncharge(struct res_counter *cnt, unsigned long val)
+{
+ unsigned long flags;
+
+ spin_lock_irqsave(&cnt->lock, flags);
+ res_counter_uncharge_locked(cnt, val);
+ spin_unlock_irqrestore(&cnt->lock, flags);
+}
+
+
+static inline unsigned long *res_counter_member(struct res_counter *cnt, int member)
+{
+ switch (member) {
+ case RES_USAGE:
+  return &cnt->usage;
+ case RES_LIMIT:
+  return &cnt->limit;
+ case RES_FAILCNT:
+  return &cnt->failcnt;
+ };
+
+ BUG();
+ return NULL;
+}
+
+ssize_t res_counter_read(struct res_counter *cnt, int member,
+  const char __user *userbuf, size_t nbytes, loff_t *pos)
+{
+ unsigned long *val;
+ char buf[64], *s;
+
+ s = buf;
+ val = res_counter_member(cnt, member);
+ s += sprintf(s, "%lu\n", *val);
+ return simple_read_from_buffer((void __user *)userbuf, nbytes,
+   pos, buf, s - buf);
+}
```

```
+
+ssize_t res_counter_write(struct res_counter *cnt, int member,
+  const char __user *userbuf, size_t nbytes, loff_t *pos)
+{
+ int ret;
+ char *buf, *end;
+ unsigned long tmp, *val;
+
+ buf = kmalloc(nbytes + 1, GFP_KERNEL);
+ ret = -ENOMEM;
+ if (buf == NULL)
+  goto out;
+
+ buf[nbytes] = 0;
+ ret = -EFAULT;
+ if (copy_from_user(buf, userbuf, nbytes))
+  goto out_free;
+
+ ret = -EINVAL;
+ tmp = simple_strtoul(buf, &end, 10);
+ if (*end != '\0')
+  goto out_free;
+
+ val = res_counter_member(cnt, member);
+ *val = tmp;
+ ret = nbytes;
+out_free:
+ kfree(buf);
+out:
+ return ret;
+}
```