Subject: [RFC][PATCH][1/4] RSS controller setup (
Posted by Balbir Singh on Sat, 24 Feb 2007 14:45:13 GMT
View Forum Message <> Reply to Message

Changelog

1. Change the name from memctlr to memcontrol
2. Coding style changes, call the API and then check return value (for kmalloc).
3. Change the output format, to print sizes in both pages and kB
4. Split the usage and limit files to be independent (cat memcontrol_usage
   no longer prints the limit)

TODO's

1. Implement error handling mechansim for handling container_add_file()
   failures (this would depend on the containers code).

This patch sets up the basic controller infrastructure on top of the
containers infrastructure. Two files are provided for monitoring
and control memcontrol_usage and memcontrol_limit.

memcontrol_usage shows the current usage (in pages, of RSS) and the limit
set by the user.

memcontrol_limit can be used to set a limit on the RSS usage of the resource.
A special value of 0, indicates that the usage is unlimited. The limit
is set in units of pages.

Signed-off-by: <balbir@in.ibm.com>
---

 include/linux/memcontrol.h | 33 +++++++
 init/Kconfig | 7 +
 mm/Makefile | 1
 mm/memcontrol.c | 193 ++
 4 files changed, 234 insertions(+)

diff -puN /dev/null include/linux/memcontrol.h
--- /dev/null 2007-02-02 22:51:23.000000000 +0530
+++ linux-2.6.20-balbir/include/linux/memcontrol.h 2007-02-24 19:39:03.000000000 +0530
@@ -0,0 +1,33 @@
+/*
+ * memcontrol.h - Memory Controller for containers
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of version 2.1 of the GNU Lesser General Public License
+ * as published by the Free Software Foundation.
+ *

```
+ * This program is distributed in the hope that it would be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software
+ * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
+ *
+ * © Copyright IBM Corporation, 2006-2007
+ *
+ * Author: Balbir Singh <balbir@in.ibm.com>
+ *
+ */
+
+#ifndef _LINUX_MEMCONTROL_H
+#define _LINUX_MEMCONTROL_H
+
+#ifdef CONFIG_CONTAINER_MEMCONTROL
+#ifndef kB
+#define kB 1024   /* One Kilo Byte */
+#endif
+
+#else /* CONFIG_CONTAINER_MEMCONTROL  */
+
+#endif /* CONFIG_CONTAINER_MEMCONTROL */
+#endif /* _LINUX_MEMCONTROL_H */
diff -puN init/Kconfig~memcontrol-setup init/Kconfig
--- linux-2.6.20/init/Kconfig~memcontrol-setup 2007-02-20 21:01:28.000000000 +0530
+++ linux-2.6.20-balbir/init/Kconfig 2007-02-20 21:01:28.000000000 +0530
@@ -306,6 +306,13 @@ config CONTAINER_NS
          for instance virtual servers and checkpoint/restart
          jobs.

+config CONTAINER_MEMCONTROL
+ bool "A simple RSS based memory controller"
+ select CONTAINERS
+ help
+   Provides a simple Resource Controller for monitoring and
+   controlling the total Resident Set Size of the tasks in a container
+
 config RELAY
  bool "Kernel->user space relay support (formerly relayfs)"
  help
diff -puN mm/Makefile~memcontrol-setup mm/Makefile
--- linux-2.6.20/mm/Makefile~memcontrol-setup 2007-02-20 21:01:28.000000000 +0530
+++ linux-2.6.20-balbir/mm/Makefile 2007-02-20 21:01:28.000000000 +0530
@@ -29,3 +29,4 @@ obj-$(CONFIG_MEMORY_HOTPLUG) += memory_h
 obj-$(CONFIG_FS_XIP) += filemap_xip.o
```

```
 obj-$(CONFIG_MIGRATION) += migrate.o
 obj-$(CONFIG_SMP) += allocpercpu.o
+obj-$(CONFIG_CONTAINER_MEMCONTROL) += memcontrol.o
diff -puN /dev/null mm/memcontrol.c
--- /dev/null 2007-02-02 22:51:23.000000000 +0530
+++ linux-2.6.20-balbir/mm/memcontrol.c 2007-02-24 19:39:24.000000000 +0530
@@ -0,0 +1,193 @@
+/*
+ * memcontrol.c - Memory Controller for containers
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of version 2.1 of the GNU Lesser General Public License
+ * as published by the Free Software Foundation.
+ *
+ * This program is distributed in the hope that it would be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software
+ * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
+ *
+ * © Copyright IBM Corporation, 2006-2007
+ *
+ * Author: Balbir Singh <balbir@in.ibm.com>
+ *
+ */
+
+#include <linux/init.h>
+#include <linux/parser.h>
+#include <linux/fs.h>
+#include <linux/container.h>
+#include <linux/memcontrol.h>
+
+#include <asm/uaccess.h>
+
+#define RES_USAGE_NO_LIMIT 0
+static const char version[] = "0.1";
+
+struct res_counter {
+ atomic_long_t usage; /* The current usage of the resource being */
+    /* counted        */
+ atomic_long_t limit; /* The limit on the resource     */
+};
+
+/*
+ * Each task belongs to a container, each container has a struct
+ * memcontrol associated with it
```

```
+ */
+struct memcontrol {
+ struct container_subsys_state css;
+ struct res_counter   counter;
+};
+
+static struct container_subsys memcontrol_subsys;
+
+static inline struct memcontrol *memcontrol_from_cont(struct container *cont)
+{
+ return container_of(container_subsys_state(cont, &memcontrol_subsys),
+    struct memcontrol, css);
+}
+
+static inline struct memcontrol *memcontrol_from_task(struct task_struct *p)
+{
+ return memcontrol_from_cont(task_container(p, &memcontrol_subsys));
+}
+
+static int memcontrol_create(struct container_subsys *ss,
+    struct container *cont)
+{
+ struct memcontrol *mem = kzalloc(sizeof(*mem), GFP_KERNEL);
+ if (!mem)
+  return -ENOMEM;
+
+ cont->subsys[memcontrol_subsys.subsys_id] = &mem->css;
+ atomic_long_set(&mem->counter.usage, 0);
+ atomic_long_set(&mem->counter.limit, 0);
+ return 0;
+}
+
+static void memcontrol_destroy(struct container_subsys *ss,
+    struct container *cont)
+{
+ kfree(memcontrol_from_cont(cont));
+}
+
+static ssize_t memcontrol_limit_write(struct container *cont,
+     struct cftype *cft, struct file *file,
+     const char __user *userbuf,
+     size_t nbytes, loff_t *ppos)
+{
+ char *buffer;
+ int ret = 0;
+ unsigned long limit;
+ struct memcontrol *mem = memcontrol_from_cont(cont);
+
```

```
+ BUG_ON(!mem);
+ buffer = kmalloc(nbytes + 1, GFP_KERNEL);
+ if (buffer == NULL)
+  return -ENOMEM;
+
+ buffer[nbytes] = 0;
+ if (copy_from_user(buffer, userbuf, nbytes)) {
+  ret = -EFAULT;
+  goto out_err;
+ }
+
+ container_manage_lock();
+ if (container_is_removed(cont)) {
+  ret = -ENODEV;
+  goto out_unlock;
+ }
+
+ limit = simple_strtoul(buffer, NULL, 10);
+ /*
+  * 0 is a valid limit (unlimited resource usage)
+  */
+ if (!limit && strcmp(buffer, "0"))
+  goto out_unlock;
+
+ atomic_long_set(&mem->counter.limit, limit);
+ ret = nbytes;
+out_unlock:
+ container_manage_unlock();
+out_err:
+ kfree(buffer);
+ return ret;
+}
+
+static ssize_t memcontrol_limit_read(struct container *cont, struct cftype *cft,
+     struct file *file, char __user *userbuf,
+     size_t nbytes, loff_t *ppos)
+{
+ long limit;
+ char buf[64];  /* Move away from stack later */
+ char *s = buf;
+ struct memcontrol *mem = memcontrol_from_cont(cont);
+
+ limit = atomic_long_read(&mem->counter.limit);
+
+ s += sprintf(s, "limit: %ld pages (%ld kB)\n", limit,
+   (limit * PAGE_SIZE) / kB);
+ return simple_read_from_buffer(userbuf, nbytes, ppos, buf, s - buf);
+}
```

```
+
+static ssize_t memcontrol_usage_read(struct container *cont, struct cftype *cft,
+     struct file *file, char __user *userbuf,
+     size_t nbytes, loff_t *ppos)
+{
+ long usage;
+ char buf[64];  /* Move away from stack later */
+ char *s = buf;
+ struct memcontrol *mem = memcontrol_from_cont(cont);
+
+ usage = atomic_long_read(&mem->counter.usage);
+
+ s += sprintf(s, "usage: %ld pages (%ld kB)\n", usage,
+   (usage * PAGE_SIZE) / kB);
+ return simple_read_from_buffer(userbuf, nbytes, ppos, buf, s - buf);
+}
+
+static struct cftype memcontrol_usage = {
+ .name = "memcontrol_usage",
+ .read = memcontrol_usage_read,
+};
+
+static struct cftype memcontrol_limit = {
+ .name = "memcontrol_limit",
+ .write = memcontrol_limit_write,
+ .read = memcontrol_limit_read,
+};
+
+static int memcontrol_populate(struct container_subsys *ss,
+    struct container *cont)
+{
+ int rc;
+ if ((rc = container_add_file(cont, &memcontrol_usage)) < 0)
+  return rc;
+ if ((rc = container_add_file(cont, &memcontrol_limit)) < 0)
+  return rc;
+ return 0;
+}
+
+static struct container_subsys memcontrol_subsys = {
+ .name = "memcontrol",
+ .create = memcontrol_create,
+ .destroy = memcontrol_destroy,
+ .populate = memcontrol_populate,
+};
+
+int __init memcontrol_init(void)
+{
```

```
+ int id;
+
+ id = container_register_subsys(&memcontrol_subsys);
+ printk("Initializing memcontrol version %s, id %d\n", version, id);
+ return id < 0 ? id : 0;
+}
+
+module_init(memcontrol_init);
```

_

--
Warm Regards,
Balbir Singh
