Subject: [-mm PATCH] Memory controller improve user interface
Posted by Balbir Singh on Wed, 29 Aug 2007 11:10:30 GMT
View Forum Message <> Reply to Message

Change the interface to use kilobytes instead of pages. Page sizes can vary
across platforms and configurations. A new strategy routine has been added
to the resource counters infrastructure to format the data as desired.

Suggested by David Rientjes, Andrew Morton and Herbert Poetzl

Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>

 Documentation/controllers/memory.txt | 7 +++--
 include/linux/res_counter.h | 6 ++--
 kernel/res_counter.c | 24 ++++++++++++----
 mm/memcontrol.c | 47 ++++++++++++++++++++++++++++++++--------
 4 files changed, 64 insertions(+), 20 deletions(-)

diff -puN mm/memcontrol.c~mem-control-make-ui-use-kilobytes mm/memcontrol.c
--- linux-2.6.23-rc3/mm/memcontrol.c~mem-control-make-ui-use-kilobytes 2007-08-28
13:20:44.000000000 +0530
+++ linux-2.6.23-rc3-balbir/mm/memcontrol.c 2007-08-29 14:36:07.000000000 +0530
@@ -32,6 +32,7 @@

 struct container_subsys mem_container_subsys;
 static const int MEM_CONTAINER_RECLAIM_RETRIES = 5;
+static const int MEM_CONTAINER_CHARGE_KB = (PAGE_SIZE >> 10);

 /*
  * The memory controller data structure. The memory controller controls both
@@ -312,7 +313,7 @@ int mem_container_charge(struct page *pa
   * If we created the page_container, we should free it on exceeding
   * the container limit.
   */
- while (res_counter_charge(&mem->res, 1)) {
+ while (res_counter_charge(&mem->res, MEM_CONTAINER_CHARGE_KB)) {
   if (try_to_free_mem_container_pages(mem))
    continue;

@@ -352,7 +353,7 @@ int mem_container_charge(struct page *pa
   kfree(pc);
   pc = race_pc;
   atomic_inc(&pc->ref_cnt);
- res_counter_uncharge(&mem->res, 1);
+ res_counter_uncharge(&mem->res, MEM_CONTAINER_CHARGE_KB);
   css_put(&mem->css);
   goto done;

```
 }
@@ -417,7 +418,7 @@ void mem_container_uncharge(struct page_
   css_put(&mem->css);
   page_assign_page_container(page, NULL);
   unlock_page_container(page);
-  res_counter_uncharge(&mem->res, 1);
+  res_counter_uncharge(&mem->res, MEM_CONTAINER_CHARGE_KB);

   spin_lock_irqsave(&mem->lru_lock, flags);
   list_del_init(&pc->lru);
@@ -426,12 +427,37 @@ void mem_container_uncharge(struct page_
 }
 }

-static ssize_t mem_container_read(struct container *cont, struct cftype *cft,
-   struct file *file, char __user *userbuf, size_t nbytes,
-   loff_t *ppos)
+int mem_container_read_strategy(unsigned long val, char *buf)
+{
+ return sprintf(buf, "%lu (kB)\n", val);
+}
+
+int mem_container_write_strategy(char *buf, unsigned long *tmp)
+{
+ *tmp = memparse(buf, &buf);
+ if (*buf != '\0')
+  return -EINVAL;
+
+ *tmp = *tmp >> 10;  /* convert to kilobytes */
+ return 0;
+}
+
+static ssize_t mem_container_read_usage(struct container *cont,
+   struct cftype *cft, struct file *file,
+   char __user *userbuf, size_t nbytes, loff_t *ppos)
+{
+ return res_counter_read(&mem_container_from_cont(cont)->res,
+    cft->private, userbuf, nbytes, ppos,
+    mem_container_read_strategy);
+}
+
+static ssize_t mem_container_read(struct container *cont,
+   struct cftype *cft, struct file *file,
+   char __user *userbuf, size_t nbytes, loff_t *ppos)
 {
  return res_counter_read(&mem_container_from_cont(cont)->res,
-    cft->private, userbuf, nbytes, ppos);
+    cft->private, userbuf, nbytes, ppos,
```

```
+    NULL);
 }

 static ssize_t mem_container_write(struct container *cont, struct cftype *cft,
@@ -439,7 +465,8 @@ static ssize_t mem_container_write(struc
     size_t nbytes, loff_t *ppos)
 {
  return res_counter_write(&mem_container_from_cont(cont)->res,
-    cft->private, userbuf, nbytes, ppos);
+    cft->private, userbuf, nbytes, ppos,
+    mem_container_write_strategy);
 }

 static ssize_t mem_control_type_write(struct container *cont,
@@ -500,13 +527,13 @@ static struct cftype mem_container_files
  {
   .name = "usage",
   .private = RES_USAGE,
-  .read = mem_container_read,
+  .read = mem_container_read_usage,
  },
  {
   .name = "limit",
   .private = RES_LIMIT,
   .write = mem_container_write,
-  .read = mem_container_read,
+  .read = mem_container_read_usage,
  },
  {
   .name = "failcnt",
diff -puN include/linux/memcontrol.h~mem-control-make-ui-use-kilobytes
include/linux/memcontrol.h
diff -puN include/linux/res_counter.h~mem-control-make-ui-use-kilobytes
include/linux/res_counter.h
--- linux-2.6.23-rc3/include/linux/res_counter.h~mem-control-make-ui-use-kilobytes 2007-08-28
13:21:36.000000000 +0530
+++ linux-2.6.23-rc3-balbir/include/linux/res_counter.h 2007-08-29 00:52:45.000000000 +0530
@@ -52,9 +52,11 @@ struct res_counter {
  */

 ssize_t res_counter_read(struct res_counter *counter, int member,
-  const char __user *buf, size_t nbytes, loff_t *pos);
+  const char __user *buf, size_t nbytes, loff_t *pos,
+  int (*read_strategy)(unsigned long val, char *s));
 ssize_t res_counter_write(struct res_counter *counter, int member,
-  const char __user *buf, size_t nbytes, loff_t *pos);
+  const char __user *buf, size_t nbytes, loff_t *pos,
+  int (*write_strategy)(char *buf, unsigned long *val));
```

```
 /*
  * the field descriptors. one for each member of res_counter
diff -puN kernel/res_counter.c~mem-control-make-ui-use-kilobytes kernel/res_counter.c
--- linux-2.6.23-rc3/kernel/res_counter.c~mem-control-make-ui-use-kilobytes 2007-08-28
13:21:41.000000000 +0530
+++ linux-2.6.23-rc3-balbir/kernel/res_counter.c 2007-08-29 14:54:04.000000000 +0530
@@ -76,20 +76,25 @@ static inline unsigned long *res_counter
 }

 ssize_t res_counter_read(struct res_counter *counter, int member,
-  const char __user *userbuf, size_t nbytes, loff_t *pos)
+  const char __user *userbuf, size_t nbytes, loff_t *pos,
+  int (*read_strategy)(unsigned long val, char *st_buf))
 {
  unsigned long *val;
  char buf[64], *s;

  s = buf;
  val = res_counter_member(counter, member);
- s += sprintf(s, "%lu\n", *val);
+ if (read_strategy)
+  s += read_strategy(*val, s);
+ else
+  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 *counter, int member,
-  const char __user *userbuf, size_t nbytes, loff_t *pos)
+  const char __user *userbuf, size_t nbytes, loff_t *pos,
+  int (*write_strategy)(char *st_buf, unsigned long *val))
 {
  int ret;
  char *buf, *end;
@@ -106,9 +111,16 @@ ssize_t res_counter_write(struct res_cou
   goto out_free;

  ret = -EINVAL;
- tmp = simple_strtoul(buf, &end, 10);
- if (*end != '\0')
-  goto out_free;
+
+ if (write_strategy) {
+  if (write_strategy(buf, &tmp)) {
+   goto out_free;
+  }
```

```
+ } else {
+  tmp = simple_strtoul(buf, &end, 10);
+  if (*end != '\0')
+   goto out_free;
+ }

  val = res_counter_member(counter, member);
  *val = tmp;
diff -puN Documentation/controllers/memory.txt~mem-control-make-ui-use-kilobytes
Documentation/controllers/memory.txt
---
linux-2.6.23-rc3/Documentation/controllers/memory.txt~mem-control-make-ui-use-kilobytes 2007-
08-29 14:42:03.000000000 +0530
+++ linux-2.6.23-rc3-balbir/Documentation/controllers/memory.txt 2007-08-29
14:50:42.000000000 +0530
@@ -165,11 +165,14 @@ c. Enable CONFIG_CONTAINER_MEM_CONT

 Since now we're in the 0 container,
 We can alter the memory limit:
-# echo -n 6000 > /containers/0/memory.limit
+# echo -n 4000K > /containers/0/memory.limit
+
+NOTE: The interface has now changed to display the usage in kilobytes
+instead of pages

 We can check the usage:
 # cat /containers/0/memory.usage
-25
+56 (kB)

 The memory.failcnt field gives the number of times that the container limit was
 exceeded.
_

--
 Warm Regards,
 Balbir Singh
 Linux Technology Center
 IBM, ISTL
_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers
```