
Subject: [PATCH] Simplify memory controller and resource counter I/O

Posted by [Paul Menage](#) on Fri, 05 Oct 2007 04:35:45 GMT

[View Forum Message](#) <> [Reply to Message](#)

Simplify the memory controller and resource counter I/O routines

This patch strips out some I/O boilerplate from resource counters and the memory controller. It also adds locking to the resource counter reads and writes, and forbids writes to the root memory cgroup's limit file.

cgroup_write_uint() is extended to call memparse() rather than just simple_strtoul()

Signed-off-by: Paul Menage <menage@google.com>

```
include/linux/cgroup.h      |  2
include/linux/res_counter.h | 13 +----
kernel/cgroup.c             |  2
kernel/res_counter.c        | 64 ++++++-----
mm/memcontrol.c            | 103 ++++++++-----
5 files changed, 45 insertions(+), 139 deletions(-)
```

Index: container-2.6.23-rc8-mm2/include/linux/res_counter.h

=====

--- container-2.6.23-rc8-mm2.orig/include/linux/res_counter.h

+++ container-2.6.23-rc8-mm2/include/linux/res_counter.h

@@ -46,17 +46,12 @@ struct res_counter {

```
 *
 * @counter:   the counter in question
 * @member:   the field to work with (see RES_xxx below)
- * @buf:     the buffer to operate on,...
- * @nbytes:  its size...
- * @pos:     and the offset.
+ * @val:     the value passed by the user (for write)
 */
```

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

```

/*
 * the field descriptors. one for each member of res_counter
Index: container-2.6.23-rc8-mm2/kernel/res_counter.c
=====
--- container-2.6.23-rc8-mm2.orig/kernel/res_counter.c
+++ container-2.6.23-rc8-mm2/kernel/res_counter.c
@@ -75,58 +75,22 @@ res_counter_member(struct res_counter *c
     return NULL;
 }

-ssize_t res_counter_read(struct res_counter *counter, int member,
-    const char __user *userbuf, size_t nbytes, loff_t *pos,
-    int (*read_strategy)(unsigned long long val, char *st_buf))
+unsigned long long res_counter_read(struct res_counter *counter, int member)
{
-    unsigned long long *val;
-    char buf[64], *s;
-
-    s = buf;
-    val = res_counter_member(counter, member);
-    if (read_strategy)
-        s += read_strategy(*val, s);
-    else
-        s += sprintf(s, "%llu\n", *val);
-    return simple_read_from_buffer((void __user *)userbuf, nbytes,
-        pos, buf, s - buf);
+    unsigned long long val;
+    unsigned long flags;
+    spin_lock_irqsave(&counter->lock, flags);
+    val = *res_counter_member(counter, member);
+    spin_unlock_irqrestore(&counter->lock, flags);
+    return val;
}

-ssize_t res_counter_write(struct res_counter *counter, int member,
-    const char __user *userbuf, size_t nbytes, loff_t *pos,
-    int (*write_strategy)(char *st_buf, unsigned long long *val))
+int res_counter_write(struct res_counter *counter, int member,
+    unsigned long long val)
{
-    int ret;
-    char *buf, *end;
-    unsigned long 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;
-
- if (write_strategy) {
- if (write_strategy(buf, &tmp)) {
- goto out_free;
- }
- } else {
- tmp = simple_strtoll(buf, &end, 10);
- if (*end != '\0')
- goto out_free;
- }
-
- val = res_counter_member(counter, member);
- *val = tmp;
- ret = nbytes;
-out_free:
- kfree(buf);
-out:
- return ret;
+ unsigned long flags;
+ spin_lock_irqsave(&counter->lock, flags);
+ *res_counter_member(counter, member) = val;
+ spin_unlock_irqrestore(&counter->lock, flags);
+ return 0;
}
Index: container-2.6.23-rc8-mm2/mm/memcontrol.c
=====
--- container-2.6.23-rc8-mm2.orig/mm/memcontrol.c
+++ container-2.6.23-rc8-mm2/mm/memcontrol.c
@@ -432,112 +432,59 @@ void mem_cgroup_uncharge(struct page_cgr
}
}

-int mem_cgroup_write_strategy(char *buf, unsigned long long *tmp)
-{
- *tmp = memparse(buf, &buf);
- if (*buf != '\0')
- return -EINVAL;
-
- /*
-  * Round up the value to the closest page size

```

```

- */
- *tmp = ((*tmp + PAGE_SIZE - 1) >> PAGE_SHIFT) << PAGE_SHIFT;
- return 0;
-}
-
-static ssize_t mem_cgroup_read(struct cgroup *cont,
- struct cftype *cft, struct file *file,
- char __user *userbuf, size_t nbytes, loff_t *ppos)
+static unsigned long long mem_cgroup_read(struct cgroup *cont,
+ struct cftype *cft)
{
    return res_counter_read(&mem_cgroup_from_cont(cont)->res,
- cft->private, userbuf, nbytes, ppos,
- NULL);
+ cft->private);
}

-static ssize_t mem_cgroup_write(struct cgroup *cont, struct cftype *cft,
- struct file *file, const char __user *userbuf,
- size_t nbytes, loff_t *ppos)
+static int mem_cgroup_write(struct cgroup *cont, struct cftype *cft,
+ unsigned long long val)
{
+ /* Don't allow the limit to be set for the root cgroup */
+ if (!cont->parent)
+ return -EINVAL;
    return res_counter_write(&mem_cgroup_from_cont(cont)->res,
- cft->private, userbuf, nbytes, ppos,
- mem_cgroup_write_strategy);
+ cft->private, PAGE_ALIGN(val));
}

-static ssize_t mem_control_type_write(struct cgroup *cont,
- struct cftype *cft, struct file *file,
- const char __user *userbuf,
- size_t nbytes, loff_t *pos)
-{
- int ret;
- char *buf, *end;
- unsigned long tmp;
- struct mem_cgroup *mem;
-
- mem = mem_cgroup_from_cont(cont);
- 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;
-
- if (tmp <= MEM_CGROUP_TYPE_UNSPEC || tmp >= MEM_CGROUP_TYPE_MAX)
- goto out_free;
-
- mem->control_type = tmp;
- ret = nbytes;
-out_free:
- kfree(buf);
-out:
- return ret;
+static int mem_control_type_write(struct cgroup *cont, struct cftype *cft,
+    u64 val)
+{
+ if (val <= MEM_CGROUP_TYPE_UNSPEC || val >= MEM_CGROUP_TYPE_MAX)
+ return -EINVAL;
+ mem_cgroup_from_cont(cont)->control_type = val;
+ return 0;
+}

-static ssize_t mem_control_type_read(struct cgroup *cont,
-    struct cftype *cft,
-    struct file *file, char __user *userbuf,
-    size_t nbytes, loff_t *ppos)
+static u64 mem_control_type_read(struct cgroup *cont,
+    struct cftype *cft)
+{
- unsigned long val;
- char buf[64], *s;
- struct mem_cgroup *mem;
-
- mem = mem_cgroup_from_cont(cont);
- s = buf;
- val = mem->control_type;
- s += sprintf(s, "%lu\n", val);
- return simple_read_from_buffer((void __user *)userbuf, nbytes,
-    ppos, buf, s - buf);
+ return mem_cgroup_from_cont(cont)->control_type;
+}

```

```

static struct cftype mem_cgroup_files[] = {
{
.name = "usage_in_bytes",
.private = RES_USAGE,
- .read = mem_cgroup_read,
+ .read_uint = mem_cgroup_read,
},
{
.name = "limit_in_bytes",
.private = RES_LIMIT,
- .write = mem_cgroup_write,
- .read = mem_cgroup_read,
+ .write_uint = mem_cgroup_write,
+ .read_uint = mem_cgroup_read,
},
{
.name = "failcnt",
.private = RES_FAILCNT,
- .read = mem_cgroup_read,
+ .read_uint = mem_cgroup_read,
},
{
.name = "control_type",
- .write = mem_control_type_write,
- .read = mem_control_type_read,
+ .write_uint = mem_control_type_write,
+ .read_uint = mem_control_type_read,
},
};

```

Index: container-2.6.23-rc8-mm2/include/linux/cgroup.h

```

=====
--- container-2.6.23-rc8-mm2.orig/include/linux/cgroup.h
+++ container-2.6.23-rc8-mm2/include/linux/cgroup.h
@@ -199,7 +199,7 @@ struct cftype {

```

```

/*
 * write_uint() is a shortcut for the common case of accepting
- * a single integer (as parsed by simple_strtoul) from
+ * a single integer (as parsed by lib/cmdline.c:memparse()) from
 * userspace. Use in place of write(); return 0 or error.
 */
int (*write_uint)(struct cgroup *cont, struct cftype *cft, u64 val);

```

Index: container-2.6.23-rc8-mm2/kernel/cgroup.c

```

=====
--- container-2.6.23-rc8-mm2.orig/kernel/cgroup.c
+++ container-2.6.23-rc8-mm2/kernel/cgroup.c
@@ -1296,7 +1296,7 @@ static ssize_t cgroup_write_uint(struct

```

```
/* strip newline if necessary */
if (nbytes && (buffer[nbytes-1] == '\n'))
    buffer[nbytes-1] = 0;
- val = simple_strtoull(buffer, &end, 0);
+ val = memparse(buffer, &end);
    if (*end)
        return -EINVAL;
```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
