
Subject: [PATCH 5/8] CGroup Files: Turn attach_task_by_pid directly into a cgroup write handler

Posted by [Paul Menage](#) on Fri, 20 Jun 2008 23:44:03 GMT

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

This patch changes attach_task_by_pid() to take a u64 rather than a string; as a result it can be called directly as a control groups write_u64 handler, and cgroup_common_file_write() can be removed.

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

kernel/cgroup.c | 80 ++++++++-----
1 file changed, 14 insertions(+), 66 deletions(-)

Index: cws-2.6.26-rc5-mm3/kernel/cgroup.c

=====

--- cws-2.6.26-rc5-mm3.orig/kernel/cgroup.c

+++ cws-2.6.26-rc5-mm3/kernel/cgroup.c

@ @ -504,10 +504,6 @ @ static struct css_set *find_css_set(

 * knows that the cgroup won't be removed, as cgroup_rmdir()

 * needs that mutex.

 *

- * The cgroup_common_file_write handler for operations that modify

- * the cgroup hierarchy holds cgroup_mutex across the entire operation,

- * single threading all such cgroup modifications across the system.

- *

 * The fork and exit callbacks cgroup_fork() and cgroup_exit(), don't

 * (usually) take cgroup_mutex. These are the two most performance

 * critical pieces of code here. The exception occurs on cgroup_exit(),

@ @ -1279,18 +1275,14 @ @ int cgroup_attach_task(struct cgroup *cg

}

/*

- * Attach task with pid 'pid' to cgroup 'cgrp'. Call with

- * cgroup_mutex, may take task_lock of task

+ * Attach task with pid 'pid' to cgroup 'cgrp'. Call with cgroup_mutex

+ * held. May take task_lock of task

*/

-static int attach_task_by_pid(struct cgroup *cgrp, char *pidbuf)

+static int attach_task_by_pid(struct cgroup *cgrp, u64 pid)

{

- pid_t pid;

 struct task_struct *tsk;

 int ret;

- if (sscanf(pidbuf, "%d", &pid) != 1)

- return -EIO;

```

-
    if (pid) {
        rcu_read_lock();
        tsk = find_task_by_vpid(pid);
@@ -1316,6 +1308,16 @@ static int attach_task_by_pid(struct cgr
        return ret;
    }

+static int cgroup_tasks_write(struct cgroup *cgrp, struct cftype *cft, u64 pid)
+{
+    int ret;
+    if (!cgroup_lock_live_group(cgrp))
+        return -ENODEV;
+    ret = attach_task_by_pid(cgrp, pid);
+    cgroup_unlock();
+    return ret;
+}
+
/* The various types of files and directories in a cgroup file system */
enum cgroup_filetype {
    FILE_ROOT,
@@ -1431,60 +1433,6 @@ static ssize_t cgroup_write_string(struc
    return retval;
}

-static ssize_t cgroup_common_file_write(struct cgroup *cgrp,
-    struct cftype *cft,
-    struct file *file,
-    const char __user *userbuf,
-    size_t nbytes, loff_t *unused_ppos)
-{
-    enum cgroup_filetype type = cft->private;
-    char *buffer;
-    int retval = 0;
-
-    if (nbytes >= PATH_MAX)
-        return -E2BIG;
-
-    /* +1 for nul-terminator */
-    buffer = kmalloc(nbytes + 1, GFP_KERNEL);
-    if (buffer == NULL)
-        return -ENOMEM;
-
-    if (copy_from_user(buffer, userbuf, nbytes)) {
-        retval = -EFAULT;
-        goto out1;
-    }
-    buffer[nbytes] = 0; /* nul-terminate */

```

```

- strstrip(buffer); /* strip -just- trailing whitespace */
-
- mutex_lock(&cgroup_mutex);
-
- /*
-  * This was already checked for in cgroup_file_write(), but
-  * check again now we're holding cgroup_mutex.
-  */
- if (cgroup_is_removed(cgrp)) {
-     retval = -ENODEV;
-     goto out2;
- }
-
- switch (type) {
- case FILE_TASKLIST:
-     retval = attach_task_by_pid(cgrp, buffer);
-     break;
- default:
-     retval = -EINVAL;
-     goto out2;
- }
-
- if (retval == 0)
-     retval = nbytes;
-out2:
- mutex_unlock(&cgroup_mutex);
-out1:
- kfree(buffer);
- return retval;
-}
-
static ssize_t cgroup_file_write(struct file *file, const char __user *buf,
                                size_t nbytes, loff_t *ppos)
{
@@ -2262,7 +2210,7 @@ static struct cftype files[] = {
    .name = "tasks",
    .open = cgroup_tasks_open,
    .read = cgroup_tasks_read,
-   .write = cgroup_common_file_write,
+   .write_u64 = cgroup_tasks_write,
    .release = cgroup_tasks_release,
    .private = FILE_TASKLIST,
},
--

```

Containers mailing list
Containers@lists.linux-foundation.org

