
Subject: [PATCH 1/2] CFS CGroup: Code cleanup
Posted by [Paul Menage](#) on Tue, 23 Oct 2007 00:49:20 GMT
[View Forum Message](#) <> [Reply to Message](#)

Clean up some CFS CGroup code

- replace "cont" with "cgrp" in a few places in the CFS cgroup code,
- use write_uint rather than write for cpu.shares write function

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

kernel/sched.c | 51 ++++++-----
1 file changed, 17 insertions(+), 34 deletions(-)

Index: container-2.6.23-mm1/kernel/sched.c

=====

--- container-2.6.23-mm1.orig/kernel/sched.c

+++ container-2.6.23-mm1/kernel/sched.c

@@ -6936,25 +6936,25 @@ unsigned long sched_group_shares(struct
#ifdef CONFIG_FAIR_CGROUP_SCHED

```
/* return corresponding task_group object of a cgroup */  
-static inline struct task_group *cgroup_tg(struct cgroup *cont)  
+static inline struct task_group *cgroup_tg(struct cgroup *cgrp)  
{  
- return container_of(cgroup_subsys_state(cont, cpu_cgroup_subsys_id),  
-    struct task_group, css);  
+ return container_of(cgroup_subsys_state(cgrp, cpu_cgroup_subsys_id),  
+    struct task_group, css);  
}
```

```
static struct cgroup_subsys_state *  
-cpu_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)  
+cpu_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cgrp)  
{  
    struct task_group *tg;
```

```
- if (!cont->parent) {  
+ if (!cgrp->parent) {  
    /* This is early initialization for the top cgroup */  
-    init_task_group.css.cgroup = cont;  
+    init_task_group.css.cgroup = cgrp;  
    return &init_task_group.css;  
}
```

```
/* we support only 1-level deep hierarchical scheduler atm */  
- if (cont->parent->parent)
```

```

+ if (cgrp->parent->parent)
    return ERR_PTR(-EINVAL);

    tg = sched_create_group();
@@ -6962,21 +6962,21 @@ cpu_cgroup_create(struct cgroup_subsys *
    return ERR_PTR(-ENOMEM);

    /* Bind the cgroup to task_group object we just created */
- tg->css.cgroup = cont;
+ tg->css.cgroup = cgrp;

    return &tg->css;
}

static void cpu_cgroup_destroy(struct cgroup_subsys *ss,
-    struct cgroup *cont)
+    struct cgroup *cgrp)
{
- struct task_group *tg = cgroup_tg(cont);
+ struct task_group *tg = cgroup_tg(cgrp);

    sched_destroy_group(tg);
}

static int cpu_cgroup_can_attach(struct cgroup_subsys *ss,
-    struct cgroup *cont, struct task_struct *tsk)
+    struct cgroup *cgrp, struct task_struct *tsk)
{
    /* We don't support RT-tasks being in separate groups */
    if (tsk->sched_class != &fair_sched_class)
@@ -6986,38 +6986,21 @@ static int cpu_cgroup_can_attach(struct
}

static void
-cpu_cgroup_attach(struct cgroup_subsys *ss, struct cgroup *cont,
+cpu_cgroup_attach(struct cgroup_subsys *ss, struct cgroup *cgrp,
    struct cgroup *old_cont, struct task_struct *tsk)
{
    sched_move_task(tsk);
}

-static ssize_t cpu_shares_write(struct cgroup *cont, struct cftype *cftype,
-    struct file *file, const char __user *userbuf,
-    size_t nbytes, loff_t *ppos)
+static int cpu_shares_write_uint(struct cgroup *cgrp, struct cftype *cftype,
+    u64 shareval)
{
- unsigned long shareval;

```

```

- struct task_group *tg = cgroup_tg(cont);
- char buffer[2*sizeof(unsigned long) + 1];
- int rc;
-
- if (nbytes > 2*sizeof(unsigned long)) /* safety check */
- return -E2BIG;
-
- if (copy_from_user(buffer, userbuf, nbytes))
- return -EFAULT;
-
- buffer[nbytes] = 0; /* nul-terminate */
- shareval = simple_strtoul(buffer, NULL, 10);
-
- rc = sched_group_set_shares(tg, shareval);
-
- return (rc < 0 ? rc : nbytes);
+ return sched_group_set_shares(cgroup_tg(cgrp), shareval);
}

-static u64 cpu_shares_read_uint(struct cgroup *cont, struct cftype *cft)
+static u64 cpu_shares_read_uint(struct cgroup *cgrp, struct cftype *cft)
{
- struct task_group *tg = cgroup_tg(cont);
+ struct task_group *tg = cgroup_tg(cgrp);

    return (u64) tg->shares;
}

```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
