
Subject: [RFC][PATCH 7/7] CGroup API: Update cpuset to use cgroup structured file API

Posted by [Paul Menage](#) on Fri, 15 Feb 2008 20:44:25 GMT

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

Many of the cpuset control files are simple integer values, which don't require the overhead of memory allocations for reads and writes.

Move the handlers for these control files into `cpuset_read_uint()` and `cpuset_write_uint()`. This also has the advantage that the control files show up as "u64" rather than "string" in the `cgroup.api` file.

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

kernel/cpuset.c | 158 ++++++-----
1 file changed, 83 insertions(+), 75 deletions(-)

Index: cgroupmap-2.6.24-mm1/kernel/cpuset.c

=====

--- cgroupmap-2.6.24-mm1.orig/kernel/cpuset.c

+++ cgroupmap-2.6.24-mm1/kernel/cpuset.c

@@ -999,19 +999,6 @@ int current_cpuset_is_being_rebound(void
{

/*

- * Call with cgroup_mutex held.

- */

-
-static int update_memory_pressure_enabled(struct cpuset *cs, char *buf)

{

- if (simple_strtoul(buf, NULL, 10) != 0)

- cpuset_memory_pressure_enabled = 1;

- else

- cpuset_memory_pressure_enabled = 0;

- return 0;

-}

-

-/*

* update_flag - read a 0 or a 1 in a file and update associated flag

* bit: the bit to update (CS_CPU_EXCLUSIVE, CS_MEM_EXCLUSIVE,

* CS_SCHED_LOAD_BALANCE,

@@ -1023,15 +1010,13 @@ static int update_memory_pressure_enable

* Call with cgroup_mutex held.

*/

-static int update_flag(cpuset_flagbits_t bit, struct cpuset *cs, char *buf)

+static int update_flag(cpuset_flagbits_t bit, struct cpuset *cs,

```

+     int turning_on)
+ {
- int turning_on;
  struct cpuset trialcs;
  int err;
  int cpus_nonempty, balance_flag_changed;

- turning_on = (simple_strtoul(buf, NULL, 10) != 0);
-
  trialcs = *cs;
  if (turning_on)
    set_bit(bit, &trialcs.flags);
@@ -1247,44 +1232,66 @@ static ssize_t cpuset_common_file_write(
  case FILE_MEMLIST:
    retval = update_nodemask(cs, buffer);
    break;
+ default:
+   retval = -EINVAL;
+   goto out2;
+ }
+
+ if (retval == 0)
+   retval = nbytes;
+out2:
+ cgroup_unlock();
+out1:
+ kfree(buffer);
+ return retval;
+}
+
+static int cpuset_write_uint(struct cgroup *cgrp, struct cftype *cft, u64 val)
+{
+ int retval = 0;
+ struct cpuset *cs = cgroup_cs(cgrp);
+ cpuset_filetype_t type = cft->private;
+
+ cgroup_lock();
+
+ if (cgroup_is_removed(cgrp)) {
+   cgroup_unlock();
+   return -ENODEV;
+ }
+
+ switch (type) {
+   case FILE_CPU_EXCLUSIVE:
-   retval = update_flag(CS_CPU_EXCLUSIVE, cs, buffer);
+   retval = update_flag(CS_CPU_EXCLUSIVE, cs, val);
    break;

```

```

case FILE_MEM_EXCLUSIVE:
- retval = update_flag(CS_MEM_EXCLUSIVE, cs, buffer);
+ retval = update_flag(CS_MEM_EXCLUSIVE, cs, val);
  break;
case FILE_SCHED_LOAD_BALANCE:
- retval = update_flag(CS_SCHED_LOAD_BALANCE, cs, buffer);
+ retval = update_flag(CS_SCHED_LOAD_BALANCE, cs, val);
  break;
case FILE_MEMORY_MIGRATE:
- retval = update_flag(CS_MEMORY_MIGRATE, cs, buffer);
+ retval = update_flag(CS_MEMORY_MIGRATE, cs, val);
  break;
case FILE_MEMORY_PRESSURE_ENABLED:
- retval = update_memory_pressure_enabled(cs, buffer);
+ cpuset_memory_pressure_enabled = val;
  break;
case FILE_MEMORY_PRESSURE:
  retval = -EACCES;
  break;
case FILE_SPREAD_PAGE:
- retval = update_flag(CS_SPREAD_PAGE, cs, buffer);
+ retval = update_flag(CS_SPREAD_PAGE, cs, val);
  cs->mems_generation = cpuset_mems_generation++;
  break;
case FILE_SPREAD_SLAB:
- retval = update_flag(CS_SPREAD_SLAB, cs, buffer);
+ retval = update_flag(CS_SPREAD_SLAB, cs, val);
  cs->mems_generation = cpuset_mems_generation++;
  break;
default:
  retval = -EINVAL;
- goto out2;
+ break;
}
-
- if (retval == 0)
- retval = nbytes;
-out2:
  cgroup_unlock();
-out1:
- kfree(buffer);
- return retval;
+ return -EINVAL;
}

/*
@@ -1345,30 +1352,6 @@ static ssize_t cpuset_common_file_read(s
case FILE_MEMLIST:

```

```

    s += cpuset_sprintf_memlist(s, cs);
    break;
- case FILE_CPU_EXCLUSIVE:
- *s++ = is_cpu_exclusive(cs) ? '1' : '0';
- break;
- case FILE_MEM_EXCLUSIVE:
- *s++ = is_mem_exclusive(cs) ? '1' : '0';
- break;
- case FILE_SCHED_LOAD_BALANCE:
- *s++ = is_sched_load_balance(cs) ? '1' : '0';
- break;
- case FILE_MEMORY_MIGRATE:
- *s++ = is_memory_migrate(cs) ? '1' : '0';
- break;
- case FILE_MEMORY_PRESSURE_ENABLED:
- *s++ = cpuset_memory_pressure_enabled ? '1' : '0';
- break;
- case FILE_MEMORY_PRESSURE:
- s += sprintf(s, "%d", fmeter_getrate(&cs->fmeter));
- break;
- case FILE_SPREAD_PAGE:
- *s++ = is_spread_page(cs) ? '1' : '0';
- break;
- case FILE_SPREAD_SLAB:
- *s++ = is_spread_slab(cs) ? '1' : '0';
- break;
    default:
        retval = -EINVAL;
        goto out;
@@ -1381,8 +1364,32 @@ out:
    return retval;
}

-
-
+static u64 cpuset_read_uint(struct cgroup *cont, struct cftype *cft)
+{
+ struct cpuset *cs = cgroup_cs(cont);
+ cpuset_filetype_t type = cft->private;
+ switch (type) {
+ case FILE_CPU_EXCLUSIVE:
+ return is_cpu_exclusive(cs);
+ case FILE_MEM_EXCLUSIVE:
+ return is_mem_exclusive(cs);
+ case FILE_SCHED_LOAD_BALANCE:
+ return is_sched_load_balance(cs);
+ case FILE_MEMORY_MIGRATE:
+ return is_memory_migrate(cs);

```

```

+ case FILE_MEMORY_PRESSURE_ENABLED:
+ return cpuset_memory_pressure_enabled;
+ case FILE_MEMORY_PRESSURE:
+ return fmeter_getrate(&cs->fmeter);
+ break;
+ case FILE_SPREAD_PAGE:
+ return is_spread_page(cs);
+ case FILE_SPREAD_SLAB:
+ return is_spread_slab(cs);
+ default:
+ BUG();
+ }
+}

/*
@@ -1405,57 +1412,58 @@ static struct cftype cft_mems = {

static struct cftype cft_cpu_exclusive = {
    .name = "cpu_exclusive",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
    .private = FILE_CPU_EXCLUSIVE,
};

static struct cftype cft_mem_exclusive = {
    .name = "mem_exclusive",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
    .private = FILE_MEM_EXCLUSIVE,
};

static struct cftype cft_sched_load_balance = {
    .name = "sched_load_balance",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
    .private = FILE_SCHED_LOAD_BALANCE,
};

static struct cftype cft_memory_migrate = {
    .name = "memory_migrate",
- .read = cpuset_common_file_read,

```

```
- .write = cpuset_common_file_write,  
+ .read_uint = cpuset_read_uint,  
+ .read_uint = cpuset_read_uint,  
+ .write_uint = cpuset_write_uint,  
  .private = FILE_MEMORY_MIGRATE,  
};
```

```
static struct cftype cft_memory_pressure_enabled = {  
  .name = "memory_pressure_enabled",  
- .read = cpuset_common_file_read,  
- .write = cpuset_common_file_write,  
+ .read_uint = cpuset_read_uint,  
+ .write_uint = cpuset_write_uint,  
  .private = FILE_MEMORY_PRESSURE_ENABLED,  
};
```

```
static struct cftype cft_memory_pressure = {  
  .name = "memory_pressure",  
- .read = cpuset_common_file_read,  
- .write = cpuset_common_file_write,  
+ .read_uint = cpuset_read_uint,  
+ .write_uint = cpuset_write_uint,  
  .private = FILE_MEMORY_PRESSURE,  
};
```

```
static struct cftype cft_spread_page = {  
  .name = "memory_spread_page",  
- .read = cpuset_common_file_read,  
- .write = cpuset_common_file_write,  
+ .read_uint = cpuset_read_uint,  
+ .write_uint = cpuset_write_uint,  
  .private = FILE_SPREAD_PAGE,  
};
```

```
static struct cftype cft_spread_slab = {  
  .name = "memory_spread_slab",  
- .read = cpuset_common_file_read,  
- .write = cpuset_common_file_write,  
+ .read_uint = cpuset_read_uint,  
+ .write_uint = cpuset_write_uint,  
  .private = FILE_SPREAD_SLAB,  
};
```

```
@@ -1584,7 +1592,7 @@ static void cpuset_destroy(struct cgroup  
  cpuset_update_task_memory_state());
```

```
if (is_sched_load_balance(cs))  
- update_flag(CS_SCHED_LOAD_BALANCE, cs, "0");
```

```
+ update_flag(CS_SCHED_LOAD_BALANCE, cs, 0);
```

```
number_of_cpusets--;  
kfree(cs);
```

```
--
```

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