Subject: Re: [PATCH 03/33] task containersv11 add tasks file interface
Posted by Paul Jackson on Wed, 03 Oct 2007 08:09:23 GMT
View Forum Message <> Reply to Message

Cgroup (aka container) code review:

Except for the very last item below, my other comments are minor.
And the last item is pretty easy too - just more important.

Overall - nice stuff. I like this generalization of the cpuset
hierarchy. Thanks.

==========

Review comments on Documentation/cgroups.txt

- You can drop the cpuset mini-history at the top of cgroups.txt:
Modified by Paul Jackson <pj@sgi.com>
Modified by Christoph Lameter <clameter@sgi.com>

- The "Definitions" section is well done, thanks.

- Could you change the present tense in:
There are multiple efforts to provide ... These all require ...
to past tense:
There have been multiple efforts to provide ... These all required ...

The present tense suggests to me that there are, as of the time
I am reading this (perhaps years in the future) implementations of
cpusets, CKRM/ResGroups, UserBeanCounters, and virtual server
namespaces all somewhere in the main line kernel. I don't think so.

- I guess that the items "CPU", "Memory", Disk", and "Network" in
the example are separate hierarchies - but that's not clear.
Perhaps "CPU" and "Memory" are in the same hierarchy - cpusets ??
There is a mix of terminology in this example that is confusing.
The word "class" is out of context. I guess that "the WWW Network
class" refers to the "WWW browsing" thingie -- is that a cgroup --
but not to the "Network:" thingie -- is that a cgroup hierarchy ?
I got lost in this "example of a scenario ...". It reads like
a cut+paste from some other effort, incompletely reworded.

- What are these apparent 'exec notifications' that are provided to
user space that the following mentions - I cannot find any other
mention of them:

With the ability to classify tasks differently for different
resources (by putting those resource subsystems in different

hierarchies) then the admin can easily set up a script which
receives exec notifications and depending on who is launching
the browser he can

- Try replacing the following with a single period character - it's
  intruding on some description of a network "resource class" (whatever
  that is), and ends in a trailing comma:

  OR give one of the students simulation
apps enhanced CPU power,

- Could you add a two or three line example 'cgroup' following:

Each task under /proc has an added file named 'cgroup' displaying,
for each active hierarchy, the subsystem names and the cgroup name
as the path relative to the root of the cgroup file system.

- I don't see any documentation of the "/proc/cgroups" file. And for
  that matter, the format of this cgroups file is not "self-documenting"
  and neither just one value nor even an array of <key, value> pairs.

  Here's a sample /proc/cgroups file from my test system:

```
# cat /proc/cgroups
Hierarchies:
e000003015400000: bits=1 cgroups=2 (cpuset) s_active=2
a00000010118f520: bits=0 cgroups=1 ()
Subsystems:
0: name=cpuset hierarchy=e000003015400000
1: name=cpuacct hierarchy=a00000010118f520
2: name=debug hierarchy=a00000010118f520
3: name=ns hierarchy=a00000010118f520
4: name=memory hierarchy=a00000010118f520
Control Group groups: 2
```

  Can this file be both simplified and documented?

- It states in cgroups.txt:

   *** notify_on_release is disabled in the current patch set. It will be
   *** reactivated in a future patch in a less-intrusive manner

   This doesn't seem to be true, and had better not be true.
   From what I can tell, notify_on_release still works for cpusets,
   and it is important that it continue to work when cgroups are
   folded into the main line kernel.

- Typo in:

To mount a cgroup hierarchy will all available subsystems,

Should be:

To mount a cgroup hierarchy with all available subsystems,

- In the following:

Each cgroup object created by the system has an array of pointers,
indexed by subsystem id; this pointer is entirely managed by the
subsystem; the generic cgroup code will never touch this pointer.

Is plural "pointers", or singular "pointer", the correct wording?

- The following seems confused - I'd guess that all mentions of
'callback_mutex' in cgroup code and comments should be removed.

Subsystems can take/release the cgroup_mutex via the functions
cgroup_lock()/cgroup_unlock(), and can
take/release the callback_mutex via the functions
cgroup_lock()/cgroup_unlock().

- Several lines near the end of cgroups.txt start with "LL".
I guess they list what locks are held while taking the call,
but the notation seems cryptic and unfamiliar to me, and its
meaning here undocumented.

- Could you indent something (either the function declarations
or the commentary) under "3.3 Subsystem API", to make it more
readable?

- There are many instances of the local variable 'cont', referring
to a struct cgroup pointer. I presume the spelling 'cont' is a
holdover from the time when we called these containers. Perhaps
some other spelling, such as 'cgp', would be less obscure now.

==========

Review comments on include/linux/cgroup.h:

- I don't see a Google copyright - should I?

- One mention of 'callback_mutex' probably should be 'cgroup_mutex'.

- In the following:

```
/*
 * Call css_get() to hold a reference on the cgroup;
 *
 */
```

Remove extra comment line, replace ';' with '.'.

- In the following:

```
        struct cgroup *parent;  /* my parent */
```

one more tab is needed before the comment.

==========

Review comments on kernel/cgroup.c:

- Could you state at the top that 'cgroup' stands for "control group" ?

- The Google copyright is year 2006 - should that be 2006-2007 now?

- You can drop the cpuset mini-history at the top of cgroup.c:

```
 *  2003-10-10 Written by Simon Derr.
 *  2003-10-22 Updates by Stephen Hemminger.
 *  2004 May-July Rework by Paul Jackson.
```

It's still in kernel/cpuset.c, which is the more useful spot.

- Would it be a bit clearer that the define SUBSYS in kernel/cgroup.c is there for the use of the following subsys[] definition if you replaced:

```
/* Generate an array of cgroup subsystem pointers */
#define SUBSYS(_x) &_x ## _subsys,

static struct cgroup_subsys *subsys[] = {
#include <linux/cgroup_subsys.h>
};
```

with:

```
/* Generate subsys[] array of cgroup subsystem pointers */
#define SUBSYS(_x) &_x ## _subsys,
static struct cgroup_subsys *subsys[] = {
#include <linux/cgroup_subsys.h>
};
#undef SUBSYS
```

I removed a blank line, undef'd SUBSYS when done with it, and reworded the comment, so it's clear that this is a single unit

of code. On first reading it seemed that (1) the comment was claiming that the define SUBSYS itself was supposed to generate some array, which made no sense, and (2) there was no apparent use of that define SUBSYS in all of kernel/cgroup.c.

- See Documentation/CodingStyle "Section 8: Commenting" for the preferred comment style. For example, replace:

```
/* css_set_lock protects the list of css_set objects, and the
 * chain of tasks off each css_set.  Nests outside task->alloc_lock
 * due to cgroup_iter_start() */
```

with:

```
/*
 * css_set_lock protects the list of css_set objects, and the
 * chain of tasks off each css_set.  Nests outside task->alloc_lock
 * due to cgroup_iter_start()
 */
```

- The code in attach_task which skips the attachment of a task to the group it is already in has to be removed. Cpusets depends on reattaching a task to its current cpuset, in order to trigger updating the cpus_allowed mask in the task struct. This is a hack, granted, but an important one. It avoids checking for a changed cpuset 'cpus' setting in critical scheduler code paths.

  I have a patch that I will send for this change shortly, as it is more critical than the other issues I note in this review. Main line cpusets would be broken without this patch.

--

I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

_____

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