Subject: [PATCH 00/10] Containers(V10): Generic Process Containers
Posted by Paul Menage on Tue, 29 May 2007 13:01:04 GMT
View Forum Message <> Reply to Message

This is an update to my multi-hierarchy multi-subsystem generic
process containers patch. Changes since V9 (April 27th) include:

- The patchset has been rebased over 2.6.22-rc2-mm1

- A lattice of lists linking tasks to their css_groups and css_groups
to their containers has been added to support more efficient iteration
across the member tasks of a container.

- Support for the cpusets "release agent" functionality has been added
back in; this is based on a workqueue concept similar to the changes
that Cliff Wickman has been pushing for supporting CPU hot-unplug.

- Several uses of tasklist_lock replaced by reliance on RCU

- Misc cleanups

- Tested with a tweaked version of PaulJ's cpuset_test script

Still TODO:

- decide whether "Containers" is an acceptable name for the system
given its usage by some other development groups, or whether something
else (ProcessSets? ResourceGroups? TaskGroups?) would be better. I'm
inclined to leave this political decision to Andrew/Linus once they're
happy with the technical aspects of the patches.

- add a hash-table based lookup for css_group objects.

- use seq_file properly in container tasks files to avoid having to
allocate a big array for all the container's task pointers.

- lots more testing

- define standards for container file names

--

Generic Process Containers
---

There have recently been various proposals floating around for
resource management/accounting and other task grouping subsystems in
the kernel, including ResGroups, User BeanCounters, NSProxy

containers, and others.  These all need the basic abstraction of being able to group together multiple processes in an aggregate, in order to track/limit the resources permitted to those processes, or control other behaviour of the processes, and all implement this grouping in different ways.

Already existing in the kernel is the cpuset subsystem; this has a process grouping mechanism that is mature, tested, and well documented (particularly with regards to synchronization rules).

This patchset extracts the process grouping code from cpusets into a generic container system, and makes the cpusets code a client of the container system, along with a couple of simple example subsystems.

The patch set is structured as follows:

1) Basic container framework - filesystem and tracking structures

2) Simple CPU Accounting example subsystem

3) Support for the "tasks" control file

4) Hooks for fork() and exit()

5) Support for the container_clone() operation

6) Add /proc reporting interface

7) Make cpusets a container subsystem

8) Share container subsystem pointer arrays between tasks with the
   same assignments

9) Simple container debugging subsystem

10) Support for a userspace "release agent", similar to the cpusets
    release agent functionality

The intention is that the various resource management and virtualization efforts can also become container clients, with the result that:

- the userspace APIs are (somewhat) normalised

- it's easier to test out e.g. the ResGroups CPU controller in
 conjunction with the BeanCounters memory controller, or use either of them as the resource-control portion of a virtual server system.

- the additional kernel footprint of any of the competing resource
 management systems is substantially reduced, since it doesn't need
 to provide process grouping/containment, hence improving their
 chances of getting into the kernel

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

---