Subject: [PATCH 0/9] Containers (V9): Generic Process Containers
Posted by Paul Menage on Fri, 27 Apr 2007 10:46:07 GMT
View Forum Message <> Reply to Message

--

This is an update to my multi-hierarchy multi-subsystem generic
process containers patch. Changes since V8 (April 6th) include:

- The patchset has been rebased over 2.6.21-rc7-mm1

- The patchset has been restructured based on feedback; more
  functionality is now split out into separate patches where
  practical.

- The container_group structure has been renamed css_group since this
  is more descriptive of its true function

- Added a simplified file registration interface, and a simple
  interface for the common operation of returning a single number to
  userspace from a container control file

- Added a simple "debug" subsystem that is both an example of how to
  use the container system and a useful debugging tool for checking
  reference counts, etc.

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

- decide whether merging css_group and nsproxy is desirable

- 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.

- add back support for the "release agent" functionality

- 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

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>
