Subject: [PATCH 0/7] Generic Process Containers (+ ResGroups/BeanCounters)
Posted by Paul Menage on Thu, 23 Nov 2006 12:08:48 GMT
View Forum Message <> Reply to Message

This is an update to my multi-hierarchy generic containers patch (against 2.6.19-rc6). Changes include:

- an example patch implementing the BeanCounters core and numfiles counters over generic containers. The addition of the BeanCounters code unifies the three main process grouping abstractions (Cpusets, ResGroups and BeanCounters).

- a patch splitting Cpusets into two independently groupable subsystems, Cpusets and Memsets.

- support for a subsystem to keep a container alive via refcounts (e.g. the BeanCounters numfiles counter has a reference to the beancounter object from each file charged to that beancounter, so needs to be able to keep the beancounter alive until the file is destroyed)

There have recently been various proposals floating around for resource management/accounting subsystems in the kernel, including Res Groups, User BeanCounters 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, 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.

It also provides several example clients of the container system, including ResGroups and BeanCounters

The change is implemented in five stages plus two additional example patches:

1) extract the process grouping code from cpusets into a standalone system

2) remove the process grouping code from cpusets and hook into the container system

3) convert the container system to present a generic multi-hierarchy
   API, and make cpusets a client of that API

4) add a simple CPU accounting container subsystem as an example

5) example of implementing ResGroups and its numtasks controller over
   generic containers - not intended to be applied with this patch set

6) split cpusets into two subsystems, cpusets and memsets

7) example of implementing BeanCounters and its numfiles counter over
   generic containers - not intended to be applied with this patch set

The intention is that the various resource management 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

- 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>

--