Subject: [RFC][PATCH 0/7] CGroup API: More structured API for CGroups control files
Posted by Paul Menage on Fri, 15 Feb 2008 20:44:18 GMT
View Forum Message <> Reply to Message

This set of patches makes the Control Groups API more structured and self-describing.

1) Allows control files to be associated with data types such as "u64", "string", "map", etc. These types show up in a new cgroup.api file in each cgroup directory, along with a user-readable string. Files that use cgroup-provided data accessors have these file types inferred automatically.

2) Moves various files in cpusets and the memory controller from using custom-written file handlers to cgroup-defined handlers

3) Adds the "cgroup." prefix for existing cgroup-provided control files (tasks, release_agent, releasable, notify_on_release). Given than we've already had 2.6.24 go out without this prefix, I guess this could be a little contentious - but it seems like a good move to prevent name clashes in the future. (Note that this doesn't affect mounting the legacy cpuset filesystem, since the compatibility layer disables all prefixes when mounted with filesystem type "cpuset"). If people object too strongly, we could just make this the case for *new* cgroup API files, but I think this is a case where consistency would be better than compatibility - I'd be surprised if anyone has written major legacy apps yet that rely on 2.6.24 cgroup control file names.

There are various motivations for this:

1) We said at Kernel Summit '07 that the cgroup API wouldn't be allowed to spiral into an arbitrary mess of ad-hoc APIs. Having simple ways to represent common data types makes this easier. (E.g. one standard way to report a map of string,u64 pairs to userspace.)

2) People were divided on the issue of binary APIs versus ASCII APIs for control groups. Compatibility with the existing cpusets system, and ease of experimentation, were two important reasons for going with the current. ASCII API. But by having structured control files, we can open the path towards having more efficient binary APIs for simpler and more efficient programmatic access too, without any additional modifications required from the subsystems themselves.

My plans for this potential binary API are a little hazy at this point, but they might go something like opening a cgroup.bin file in a cgroup directory, and writing the names of the control files that you

were interested in; then a read on that file handle would return the contents of the given control files in a single read in a simple binary format. (Better suggestions are welcome). Regardless, getting a good typing/structure on the control files is an important first step if we want to go in that direction.

3) The memory controller currently has files with the "_in_bytes" suffix, on the grounds that otherwise it's not obvious to a new user what they represent. By moving the description to a auto-generated API file, we can remove this (IMO) inelegant suffix.

--

_____

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