Subject: Re: [PATCH 1/7] cgroup: fix and update documentation
Posted by Paul Menage on Wed, 20 Feb 2008 03:08:21 GMT
View Forum Message <> Reply to Message

On Feb 18, 2008 12:39 AM, Li Zefan <lizf@cn.fujitsu.com> wrote:
> Misc fixes and updates, make the doc consistent with current
> cgroup implementation.
>
> Signed-off-by: Li Zefan <lizf@cn.fujitsu.com>

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

Thanks for these cleanups.

Paul

> ---
> Documentation/cgroups.txt | 66 ++++++++++++++++++++++----------------------
> 1 files changed, 33 insertions(+), 33 deletions(-)
>
> diff --git a/Documentation/cgroups.txt b/Documentation/cgroups.txt
> index 42d7c4c..31d12e2 100644
>
> --- a/Documentation/cgroups.txt
> +++ b/Documentation/cgroups.txt
> @@ -28,7 +28,7 @@ CONTENTS:
> 4. Questions
>
> 1. Control Groups
> -==========
> +=================
>
> 1.1 What are cgroups ?
> ----------------------
> @@ -143,10 +143,10 @@ proliferation of such cgroups.
>
> Also lets say that the administrator would like to give enhanced network
> access temporarily to a student's browser (since it is night and the user
> -wants to do online gaming :) OR give one of the students simulation
> +wants to do online gaming :)) OR give one of the students simulation
> apps enhanced CPU power,
>
> -With ability to write pids directly to resource classes, its just a
> +With ability to write pids directly to resource classes, it's just a
> matter of :
>
> 	# echo pid > /mnt/network/<new_class>/tasks
> @@ -227,10 +227,13 @@ Each cgroup is represented by a directory in the cgroup file system

> containing the following files describing that cgroup:
>
>   - tasks: list of tasks (by pid) attached to that cgroup
> - - notify_on_release flag: run /sbin/cgroup_release_agent on exit?
> + - releasable flag: cgroup currently removeable?
> + - notify_on_release flag: run the release agent on exit?
> + - release_agent: the path to use for release notifications (this file
> +   exists in the top cgroup only)
>
> Other subsystems such as cpusets may add additional files in each
> -cgroup dir
> +cgroup dir.
>
> New cgroups are created using the mkdir system call or shell
> command. The properties of a cgroup, such as its flags, are
> @@ -257,7 +260,7 @@ performance.
> To allow access from a cgroup to the css_sets (and hence tasks)
> that comprise it, a set of cg_cgroup_link objects form a lattice;
> each cg_cgroup_link is linked into a list of cg_cgroup_links for
> -a single cgroup on its cont_link_list field, and a list of
> +a single cgroup on its cgrp_link_list field, and a list of
> cg_cgroup_links for a single css_set on its cg_link_list.
>
> Thus the set of tasks in a cgroup can be listed by iterating over
> @@ -271,9 +274,6 @@ for cgroups, with a minimum of additional kernel code.
> 1.4 What does notify_on_release do ?
> ------------------------------------
>
> -*** notify_on_release is disabled in the current patch set. It will be
> -*** reactivated in a future patch in a less-intrusive manner
> -
> If the notify_on_release flag is enabled (1) in a cgroup, then
> whenever the last task in the cgroup leaves (exits or attaches to
> some other cgroup) and the last child cgroup of that cgroup
> @@ -360,8 +360,8 @@ Now you want to do something with this cgroup.
>
> In this directory you can find several files:
> # ls
> -notify_on_release release_agent tasks
> -(plus whatever files are added by the attached subsystems)
> +notify_on_release releasable tasks
> +(plus whatever files added by the attached subsystems)
>
> Now attach your shell to this cgroup:
> # /bin/echo $$ > tasks
> @@ -404,19 +404,13 @@ with a subsystem id which will be assigned by the cgroup system.
> Other fields in the cgroup_subsys object include:
>

> - subsys_id: a unique array index for the subsystem, indicating which
> - entry in cgroup->subsys[] this subsystem should be
> - managing. Initialized by cgroup_register_subsys(); prior to this
> - it should be initialized to -1
> + entry in cgroup->subsys[] this subsystem should be managing.
>
> -- hierarchy: an index indicating which hierarchy, if any, this
> - subsystem is currently attached to. If this is -1, then the
> - subsystem is not attached to any hierarchy, and all tasks should be
> - considered to be members of the subsystem's top_cgroup. It should
> - be initialized to -1.
> +- name: should be initialized to a unique subsystem name. Should be
> + no longer than MAX_CGROUP_TYPE_NAMELEN.
>
> -- name: should be initialized to a unique subsystem name prior to
> - calling cgroup_register_subsystem. Should be no longer than
> - MAX_CGROUP_TYPE_NAMELEN
> +- early_init: indicate if the subsystem needs early initialization
> + at system boot.
>
> Each cgroup object created by the system has an array of pointers,
> indexed by subsystem id; this pointer is entirely managed by the
> @@ -434,8 +428,6 @@ situation.
> See kernel/cgroup.c for more details.
>
> 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().
>
> Accessing a task's cgroup pointer may be done in the following ways:
> @@ -444,7 +436,7 @@ Accessing a task's cgroup pointer may be done in the following ways:
> - inside an rcu_read_lock() section via rcu_dereference()
>
> 3.3 Subsystem API
> --------------------------
> +-----------------
>
> Each subsystem should:
>
> @@ -455,7 +447,8 @@ Each subsystem may export the following methods. The only
mandatory
> methods are create/destroy. Any others that are null are presumed to
> be successful no-ops.
>
> -struct cgroup_subsys_state *create(struct cgroup *cont)
> +struct cgroup_subsys_state *create(struct cgroup_subsys *ss,
> +   struct cgroup *cgrp)

> (cgroup_mutex held by caller)
>
> Called to create a subsystem state object for a cgroup. The
> @@ -470,7 +463,7 @@ identified by the passed cgroup object having a NULL parent (since
> it's the root of the hierarchy) and may be an appropriate place for
> initialization code.
>
> -void destroy(struct cgroup *cont)
> +void destroy(struct cgroup_subsys *ss, struct cgroup *cgrp)
> (cgroup_mutex held by caller)
>
> The cgroup system is about to destroy the passed cgroup; the subsystem
> @@ -481,7 +474,14 @@ cgroup->parent is still valid. (Note - can also be called for a
> newly-created cgroup if an error occurs after this subsystem's
> create() method has been called for the new cgroup).
>
> -int can_attach(struct cgroup_subsys *ss, struct cgroup *cont,
> +void pre_destroy(struct cgroup_subsys *ss, struct cgroup *cgrp);
> +(cgroup_mutex held by caller)
> +
> +Called before checking the reference count on each subsystem. This may
> +be useful for subsystems which have some extra references even if
> +there are not tasks in the cgroup.
> +
> +int can_attach(struct cgroup_subsys *ss, struct cgroup *cgrp,
> struct task_struct *task)
> (cgroup_mutex held by caller)
>
> @@ -492,8 +492,8 @@ unspecified task can be moved into the cgroup. Note that this isn't
> called on a fork. If this method returns 0 (success) then this should
> remain valid while the caller holds cgroup_mutex.
>
> -void attach(struct cgroup_subsys *ss, struct cgroup *cont,
> - struct cgroup *old_cont, struct task_struct *task)
> +void attach(struct cgroup_subsys *ss, struct cgroup *cgrp,
> + struct cgroup *old_cgrp, struct task_struct *task)
>
> Called after the task has been attached to the cgroup, to allow any
> post-attachment activity that requires memory allocations or blocking.
> @@ -505,9 +505,9 @@ registration for all existing tasks.
>
> void exit(struct cgroup_subsys *ss, struct task_struct *task)
>
> -Called during task exit
> +Called during task exit.
>
> -int populate(struct cgroup_subsys *ss, struct cgroup *cont)
> +int populate(struct cgroup_subsys *ss, struct cgroup *cgrp)

>
> Called after creation of a cgroup to allow a subsystem to populate
> the cgroup directory with file entries. The subsystem should make
> @@ -516,7 +516,7 @@ include/linux/cgroup.h for details). Note that although this
>
> method can return an error code, the error code is currently not
> always handled well.
>
> -void post_clone(struct cgroup_subsys *ss, struct cgroup *cont)
> +void post_clone(struct cgroup_subsys *ss, struct cgroup *cgrp)
>
> Called at the end of cgroup_clone() to do any paramater
> initialization which might be required before a task could attach. For
> --
> 1.5.4.rc3
>
>

_____

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