Subject: Re: [RFC/PATCH 1/8]: CGroup Files: Add locking mode to cgroups control files

Posted by Li Zefan on Tue, 13 May 2008 09:23:29 GMT

View Forum Message <> Reply to Message

menage@google.com wrote:

```
> Different cgroup files have different stability requirements of the
> cgroups framework while the handler is running; currently most
> subsystems that don't have their own internal synchronization just
> call cgroup_lock()/cgroup_unlock(), which takes the global cgroup_mutex.
>
> This patch introduces a range of locking modes that can be requested
> by a control file; currently these are all implemented internally by
> taking cgroup_mutex, but expressing the intention will make it simpler
> to move to a finer-grained locking scheme in the future.
>
> Signed-off-by: Paul Menage<menage@google.com>
>
```

This patch series looks good to me. I've reviewed those patches and didn't see anything wrong, except a little comments below.

[..snap..]

```
> -static ssize_t cgroup_write_X64(struct cgroup *cgrp, struct cftype *cft,
> -    struct file *file,
> -    const char __user *userbuf,
> -    size_t nbytes, loff_t *unused_ppos)
> +
> +
> +/**
> + * cgroup_file_lock(). Helper for cgroup read/write methods.
> + * @cgrp:  the cgroup being acted on
> + * @cft:   the control file being written to or read from
> + * *write: true if the access is a write access.
```

s/*write/@write

[..snip..]

```
>
>  static ssize_t cgroup_common_file_read(struct cgroup *cgrp,
> @@ -1518,16 +1580,21 @@ static ssize_t cgroup_file_read(struct f
>   struct cftype *cft = __d_cft(file->f_dentry);
>   struct cgroup *cgrp = __d_cgrp(file->f_dentry->d_parent);
>
> - if (!cft || cgroup_is_removed(cgrp))
> + if (cgroup_is_removed(cgrp))
>    return -ENODEV;
>
```

This check seems redundant now.

```
> - if (cft->read)
> -  return cft->read(cgrp, cft, file, buf, nbytes, ppos);
> - if (cft->read_u64)
> -  return cgroup_read_u64(cgrp, cft, file, buf, nbytes, ppos);
> - if (cft->read_s64)
> -  return cgroup_read_s64(cgrp, cft, file, buf, nbytes, ppos);
> - return -EINVAL;
> + if (cft->read) {
> +  /* Raw read function - no extra processing by cgroups */
> +  ssize_t retval = cgroup_file_lock(cgrp, cft, 0);
> +  if (!retval)
> +   retval = cft->read(cgrp, cft, file, buf, nbytes, ppos);
> +  cgroup_file_unlock(cgrp, cft, 0);
> +  return retval;
> + }
> + if (cft->read_u64 || cft->read_s64)
> +  return cgroup_read_X64(cgrp, cft, file, buf, nbytes, ppos);
> + else
> +  return -EINVAL;
>  }
```