
Subject: Re: [PATCH v4 08/14] res_counter: return amount of charges after res_counter_uncharge

Posted by [Michal Hocko](#) on Wed, 10 Oct 2012 11:25:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Mon 08-10-12 14:06:14, Glauber Costa wrote:

> It is useful to know how many charges are still left after a call to
> res_counter_uncharge. While it is possible to issue a res_counter_read
> after uncharge, this can be racy.

>

> If we need, for instance, to take some action when the counters drop
> down to 0, only one of the callers should see it. This is the same
> semantics as the atomic variables in the kernel.

>

> Since the current return value is void, we don't need to worry about
> anything breaking due to this change: nobody relied on that, and only
> users appearing from now on will be checking this value.

>

> Signed-off-by: Glauber Costa <glommer@parallels.com>

> CC: Michal Hocko <mhocko@suse.cz>

> CC: Johannes Weiner <hannes@cmpxchg.org>

> CC: Suleiman Souhlal <suleiman@google.com>

> CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

Reviewed-by: Michal Hocko <mhocko@suse.cz>

> ---

> Documentation/cgroups/resource_counter.txt | 7 ++++---

> include/linux/res_counter.h | 12 ++++++-----

> kernel/res_counter.c | 20 ++++++++-----

> 3 files changed, 24 insertions(+), 15 deletions(-)

>

> diff --git a/Documentation/cgroups/resource_counter.txt

b/Documentation/cgroups/resource_counter.txt

> index 0c4a344..c4d99ed 100644

> --- a/Documentation/cgroups/resource_counter.txt

> +++ b/Documentation/cgroups/resource_counter.txt

> @@ -83,16 +83,17 @@ to work with it.

> res_counter->lock internally (it must be called with res_counter->lock
> held). The force parameter indicates whether we can bypass the limit.

>

> - e. void res_counter_uncharge[_locked]

> + e. u64 res_counter_uncharge[_locked]

> (struct res_counter *rc, unsigned long val)

>

> When a resource is released (freed) it should be de-accounted
> from the resource counter it was accounted to. This is called

> - "uncharging".

```

> + "uncharging". The return value of this function indicate the amount
> + of charges still present in the counter.
>
> The _locked routines imply that the res_counter->lock is taken.
>
> - f. void res_counter_uncharge_until
> + f. u64 res_counter_uncharge_until
>   (struct res_counter *rc, struct res_counter *top,
>    unsigned long val)
>
> diff --git a/include/linux/res_counter.h b/include/linux/res_counter.h
> index 7d7fbe2..4b173b6 100644
> --- a/include/linux/res_counter.h
> +++ b/include/linux/res_counter.h
> @@ -130,14 +130,16 @@ int res_counter_charge_nofail(struct res_counter *counter,
>  *
>  * these calls check for usage underflow and show a warning on the console
>  * _locked call expects the counter->lock to be taken
> + *
> + * returns the total charges still present in @counter.
>  */
>
> -void res_counter_uncharge_locked(struct res_counter *counter, unsigned long val);
> -void res_counter_uncharge(struct res_counter *counter, unsigned long val);
> +u64 res_counter_uncharge_locked(struct res_counter *counter, unsigned long val);
> +u64 res_counter_uncharge(struct res_counter *counter, unsigned long val);
>
> -void res_counter_uncharge_until(struct res_counter *counter,
> -   struct res_counter *top,
> -   unsigned long val);
> +u64 res_counter_uncharge_until(struct res_counter *counter,
> +   struct res_counter *top,
> +   unsigned long val);
> /**
>  * res_counter_margin - calculate chargeable space of a counter
>  * @cnt: the counter
> diff --git a/kernel/res_counter.c b/kernel/res_counter.c
> index ad581aa..7b3d6dc 100644
> --- a/kernel/res_counter.c
> +++ b/kernel/res_counter.c
> @@ -86,33 +86,39 @@ int res_counter_charge_nofail(struct res_counter *counter, unsigned
long val,
>  return __res_counter_charge(counter, val, limit_fail_at, true);
>  }
>
> -void res_counter_uncharge_locked(struct res_counter *counter, unsigned long val)
> +u64 res_counter_uncharge_locked(struct res_counter *counter, unsigned long val)
>  {

```

```

> if (WARN_ON(counter->usage < val))
>   val = counter->usage;
>
>   counter->usage -= val;
> + return counter->usage;
> }
>
> -void res_counter_uncharge_until(struct res_counter *counter,
> -   struct res_counter *top,
> -   unsigned long val)
> +u64 res_counter_uncharge_until(struct res_counter *counter,
> +   struct res_counter *top,
> +   unsigned long val)
> {
>   unsigned long flags;
>   struct res_counter *c;
> + u64 ret = 0;
>
>   local_irq_save(flags);
>   for (c = counter; c != top; c = c->parent) {
> +   u64 r;
>   spin_lock(&c->lock);
> -   res_counter_uncharge_locked(c, val);
> +   r = res_counter_uncharge_locked(c, val);
> +   if (c == counter)
> +     ret = r;
>   spin_unlock(&c->lock);
> }
>   local_irq_restore(flags);
> + return ret;
> }
>
> -void res_counter_uncharge(struct res_counter *counter, unsigned long val)
> +u64 res_counter_uncharge(struct res_counter *counter, unsigned long val)
> {
> -   res_counter_uncharge_until(counter, NULL, val);
> +   return res_counter_uncharge_until(counter, NULL, val);
> }
>
> static inline unsigned long long *
> --
> 1.7.11.4
>
> --
> To unsubscribe from this list: send the line "unsubscribe cgroups" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html

```

--

Michal Hocko
SUSE Labs
