Subject: Re: [RFC 5/7] use percpu_counters for res_counter usage
Posted by KAMEZAWA Hiroyuki on Fri, 30 Mar 2012 09:58:03 GMT
View Forum Message <> Reply to Message

(2012/03/30 18:33), KAMEZAWA Hiroyuki wrote:

> (2012/03/30 17:04), Glauber Costa wrote:

>
> Hmm.... this part doesn't seem very good.
> I don't think for_each_online_cpu() here will not be a way to the final win.
> Under multiple hierarchy, you may need to call for_each_online_cpu() in each level.
>
> Can't you update percpu counter's core logic to avoid using for_each_online_cpu() ?
> For example, if you know what cpus have caches, you can use that cpu mask...
>
> Memo:
> Current implementation of memcg's percpu counting is reserving usage before its real use.
> In usual, the kernel don't have to scan percpu caches and just drain caches from cpus
> reserving usages if we need to cancel reserved usages. (And it's automatically canceled
> when cpu's memcg changes.)
>
> And 'reserving' avoids caching in multi-level counters,....it updates multiple counters
> in batch and memcg core don't need to walk res_counter ancestors in fast path.
>
> Considering res_counter's characteristics
> - it has _hard_ limit
> - it can be tree and usages are propagated to ancestors
> - all ancestors has hard limit.
>
> Isn't it better to generalize 'reserving resource' model ?
> You can provide 'precise usage' to the user by some logic.
>

Ah....one more point. please see this memcg's code.
==

```
        if (nr_pages == 1 && consume_stock(memcg)) {
                /*
                 * It seems dagerous to access memcg without css_get().
                 * But considering how consume_stok works, it's not
                 * necessary. If consume_stock success, some charges
                 * from this memcg are cached on this cpu. So, we
                 * don't need to call css_get()/css_tryget() before
                 * calling consume_stock().
                 */
                rcu_read_unlock();
                goto done;
        }
```

```
		/* after here, we may be blocked. we need to get refcnt */
		if (!css_tryget(&memcg->css)) {
			rcu_read_unlock();
			goto again;
		}
```
==

Now, we do consume 'reserved' usage, we can avoid css_get(), an heavy atomic ops. You may need to move this code as

```
 rcu_read_lock()
 ....
 res_counter_charge()
 if (failure) {
  css_tryget()
  rcu_read_unlock()
 } else {
  rcu_read_unlock()
  return success;
 }
```

to compare performance. This css_get() affects performance very very much.

Thanks,
-Kame
