Subject: Re: [RFC][-mm] Memory controller hierarchy support (v1)
Posted by yamamoto on Sat, 19 Apr 2008 06:56:24 GMT
View Forum Message <> Reply to Message

> -int res_counter_charge(struct res_counter *counter, unsigned long val)
> +int res_counter_charge(struct res_counter *counter, unsigned long val,
> + struct res_counter **limit_exceeded_at)
> {
> int ret;
> unsigned long flags;
> + struct res_counter *c, *unroll_c;
>
> - spin_lock_irqsave(&counter->lock, flags);
> - ret = res_counter_charge_locked(counter, val);
> - spin_unlock_irqrestore(&counter->lock, flags);
> + *limit_exceeded_at = NULL;
> + local_irq_save(flags);
> + for (c = counter; c != NULL; c = c->parent) {
> + spin_lock(&c->lock);
> + ret = res_counter_charge_locked(c, val);
> + spin_unlock(&c->lock);
> + if (ret < 0) {
> + *limit_exceeded_at = c;
> + goto unroll;
> + }
> + }
> + local_irq_restore(flags);
> + return 0;
> +
> +unroll:
> + for (unroll_c = counter; unroll_c != c; unroll_c = unroll_c->parent) {
> + spin_lock(&unroll_c->lock);
> + res_counter_uncharge_locked(unroll_c, val);
> + spin_unlock(&unroll_c->lock);
> + }
> + local_irq_restore(flags);
> return ret;
> }

i wonder how much performance impacts this involves.

it increases the number of atomic ops per charge/uncharge and
makes the common case (success) of every charge/uncharge in a system
touch a global (ie. root cgroup's) cachelines.

> + /*
> + * Ideally we need to hold cgroup_mutex here
> + */

```
> +  list_for_each_entry_safe_from(cgroup, cgrp,
> +    &curr_cgroup->children, sibling) {
> +   struct mem_cgroup *mem_child;
> +
> +   mem_child = mem_cgroup_from_cont(cgroup);
> +   ret = try_to_free_mem_cgroup_pages(mem_child,
> +         gfp_mask);
> +   mem->last_scanned_child = mem_child;
> +   if (ret == 0)
> +    break;
> +  }
```

if i read it correctly, it makes us hit the last child again and again.

i think you want to reclaim from all cgroups under the curr_cgroup
including eg. children's children.

YAMAMOTO Takashi