Subject: Re: [PATCH 2/2] Make res_counter hierarchical
Posted by KAMEZAWA Hiroyuki on Sat, 08 Mar 2008 04:44:23 GMT
View Forum Message <> Reply to Message

On Fri, 07 Mar 2008 18:32:20 +0300
Pavel Emelyanov <xemul@openvz.org> wrote:

```
> This allows us two things basically:
>
> 1. If the subgroup has the limit higher than its parent has
>    then the one will get more memory than allowed.
> 2. When we will need to account for a resource in more than
>    one place, we'll be able to use this technics.
>
>    Look, consider we have a memory limit and swap limit. The
>    memory limit is the limit for the sum of RSS, page cache
>    and swap usage. To account for this gracefuly, we'll set
>    two counters:
>
>     res_counter mem_counter;
>     res_counter swap_counter;
>
>    attach mm to the swap one
>
>     mm->mem_cnt = &swap_counter;
>
>    and make the swap_counter be mem's child. That's it. If we
>    want hierarchical support, then the tree will look like this:
>
>    mem_counter_top
>     swap_counter_top <- mm_struct living at top
>      mem_counter_sub
>       swap_counter_sub <- mm_struct living at sub
>
```
Hmm? seems strange.

IMO, a parent's usage is just sum of all childs'.
And, historically, memory overcommit is done agaist "memory usage + swap".

How about this ?
```
    <mem_counter_top, swap_counter_top>
 <mem_counter_sub, swap_counter_sub>
 <mem_counter_sub, swap_counter_sub>
 <mem_counter_sub, swap_counter_sub>

   mem_counter_top.usage == sum of all mem_coutner_sub.usage
   swap_counter_sub.usage = sum of all swap_counter_sub.usage
```

```
> @@ -976,19 +976,22 @@ static void free_mem_cgroup_per_zone_info(struct mem_cgroup
*mem, int node)
>  static struct cgroup_subsys_state *
>  mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
>  {
> - struct mem_cgroup *mem;
> + struct mem_cgroup *mem, *parent;
>   int node;
>
>   if (unlikely((cont->parent) == NULL)) {
>    mem = &init_mem_cgroup;
>    init_mm.mem_cgroup = mem;
> - } else
> +  parent = NULL;
> + } else {
>    mem = kzalloc(sizeof(struct mem_cgroup), GFP_KERNEL);
> +  parent = mem_cgroup_from_cont(cont->parent);
> + }
>
>   if (mem == NULL)
>    return ERR_PTR(-ENOMEM);
>
> - res_counter_init(&mem->res);
> + res_counter_init(&mem->res, parent ? &parent->res : NULL);
>
```
I have no objection to add some hierarchical support to res_counter.

But we should wait to add it to mem_cgroup because we have to add
some amount of codes to handle hierarchy under mem_cgroup in reasonable way.
for example)
 - hierarchical memory reclaim
 - keeping fairness between sub memory controllers.
   etc...

Thanks,
-Kame

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