Subject: Re: [PATCH v2 02/13] memcg: Kernel memory accounting infrastructure.
Posted by Glauber Costa on Mon, 12 Mar 2012 12:38:26 GMT
View Forum Message <> Reply to Message

On 03/10/2012 12:39 AM, Suleiman Souhlal wrote:
> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
> +int
> +memcg_charge_kmem(struct mem_cgroup *memcg, gfp_t gfp, long long delta)
> +{
> + struct res_counter *fail_res;
> + struct mem_cgroup *_memcg;
> + int may_oom, ret;
> +
> + may_oom = (gfp& __GFP_WAIT)&& (gfp& __GFP_FS)&&
> + !(gfp& __GFP_NORETRY);
> +
> + ret = 0;
> +
> + _memcg = memcg;
> + if (memcg&& !mem_cgroup_test_flag(memcg,
> + MEMCG_INDEPENDENT_KMEM_LIMIT)) {
> + ret = __mem_cgroup_try_charge(NULL, gfp, delta / PAGE_SIZE,
> + &_memcg, may_oom);
> + if (ret == -ENOMEM)
> + return ret;
> + }
> +
> + if (memcg&& _memcg == memcg)
> + ret = res_counter_charge(&memcg->kmem, delta,&fail_res);
> +
> + return ret;
> +}
> +
> +void
Ok.

So I've spent most of the day today trying to come up with a way not to kill the whole performance we gain from consume_stock() by this res_counter_charge() to kmem afterwards...

You mentioned you want to still be able to bill to memcg->kmem mostly for debugging/display purposes. So we're surely not using all of the res_counter infrastructure (limiting, soft limits, etc)

I was thinking: Can't we have a percpu_counter that we use for this purpose when !kmem_independent ?

we may not even need to bloat the struct, since we can fold it into a

union with struct res_counter kmem (which is bigger than a percpu counter anyway).

We just need to be a bit more careful not to allow kmem_independent to change when we already have charges to any of them (but we need to do it anyway)
