Subject: Re: [RFC] memory controller : backgorund reclaim and avoid excessive locking [4/5] borrow resource
Posted by KAMEZAWA Hiroyuki on Mon, 18 Feb 2008 02:05:53 GMT
View Forum Message <> Reply to Message

On Mon, 18 Feb 2008 09:53:35 +0900 (JST)
yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:

> > + /* try to charge */
> > + ret = res_counter_charge(&mem->res, mem->borrow_unit);
> > + if (!ret) { /* success */
> > + *bwp += (mem->borrow_unit - size);
> > + goto out;
> > + }
> > + }
> > + spin_lock(&mem->res.lock);
> > + ret = res_counter_charge_locked(&mem->res, size);
> > + spin_unlock(&mem->res.lock);
>
> although i don't know if it matters, this retrying of charge affects failcnt.
>
Hmm, yes. But what failcnt means is not so clear anyway.

> > +static void mem_cgroup_return_and_uncharge(struct mem_cgroup *mem, int size)
> > +{
> > + unsigned long flags;
> > + int uncharge_size = 0;
> > +
> > + local_irq_save(flags);
> > + if (mem->borrow_unit) {
> > + int limit = mem->borrow_unit * 2;
> > + int cpu;
> > + s64 *bwp;
> > + cpu = smp_processor_id();
> > + bwp = &mem->stat.cpustat[cpu].count[MEM_CGROUP_STAT_BORROW];
> > + *bwp += size;
> > + if (*bwp > limit) {
> > + uncharge_size = *bwp - mem->borrow_unit;
> > + *bwp = mem->borrow_unit;
> > + }
> > + } else
> > + uncharge_size = size;
> > +
> > + if (uncharge_size) {
> > + spin_lock(&mem->res.lock);
> > + res_counter_uncharge_locked(&mem->res, size);
>

> s/size/uncharge_size/
>
Oops...will fix.

> > @@ -1109,12 +1202,29 @@ static u64 mem_throttle_read(struct cgro
> > return (u64)mem->throttle.limit;
> > }
> >
> > +static int mem_bulkratio_write(struct cgroup *cont, struct cftype *cft, u64 val)
> > +{
> > + struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
> > + int unit = val * PAGE_SIZE;
> > + if (unit > (PAGE_SIZE << (MAX_ORDER/2)))
> > + return -EINVAL;
> > + mem->borrow_unit = unit;
> > + return 0;
> > +}
>
> it seems unsafe with concurrent mem_cgroup_borrow_and_charge or
> mem_cgroup_return_and_uncharge.
>
Hmm, making this to be not configurable will be good.

Thanks,
-Kame

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