

---

Subject: Re: [RFC 2/4] memcg: high-low watermark  
Posted by KAMEZAWA Hiroyuki on Wed, 28 May 2008 00:13:30 GMT  
[View Forum Message](#) <[Reply to Message](#)

---

On Tue, 27 May 2008 21:56:17 +0530

Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> KAMEZAWA Hiroyuki wrote:  
>> Add high/low watermarks to res\_counter.  
>> \*This patch itself has no behavior changes to memory resource controller.  
>>  
>> Changelog: very old one -> this one (v1)  
>> - watarmark\_state is removed and all state check is done under lock.  
>> - changed res\_counter\_charge() interface. The only user is memory  
>> resource controller. Anyway, returning -ENOMEM here is a bit starnge.  
>> - Added watermark enable/disable flag for someone don't want watermarks.  
>> - Restarted against 2.6.25-mm1.  
>> - some subsystem which doesn't want high-low watermark can work without it.

>>  
>> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>  
>> From: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

>>  
>  
> The From: line should be the first line IIRC.  
>  
ok.

>  
>> /\*  
>> + \* watermarks. needs to keep lwmark <= hwmark <= limit.  
>> + \*/  
>> + unsigned long long hwmark;  
>> + unsigned long long lwmark;  
>> + int use\_watermark;  
>  
> Is it routine to comment this way? I prefer not to have spaces in the type and  
> the member, makes it easier for my eyes.  
>  
Hmm. will fix.

>> + \* RES\_BELOW\_LIMIT -- usage is smaller than limit, success.  
>  
> ^^^ typo  
>  
sure, will fix.

```
> > + spin_unlock_irqrestore(&counter->lock, flags);
> > + return ret;
> > +}
> > +
>
> When do we return RES_OVER_LIMIT? Are we missing that here?
>
It's Bug. Yamamoto-san pointed out and I'm now fixing.
```

```
> >   spin_lock_irqsave(&counter->lock, flags);
> > + switch (member) {
> > +   case RES_LIMIT:
> > +     if (counter->use_watermark && counter->hwmark > tmp)
> > +       goto unlock_free;
>
> We need to document such API changes in the Documentation/controllers/memory.txt
> file.
>
ok, I'll add patch for documentation. to memory.txt and res_counter.txt.
```

```
> > + break;
> > + case RES_HWMARK:
> > +   if (tmp < counter->lwmrk || tmp > counter->limit)
> > +     goto unlock_free;
> > + break;
> > + case RES_LWMARK:
> > +   if (tmp > counter->hwmark)
> > +     goto unlock_free;
> > + break;
> > + default:
> > + break;
> > + }
> >   val = res_counter_member(counter, member);
> >   *val = tmp;
> > - spin_unlock_irqrestore(&counter->lock, flags);
> >   ret = nbytes;
> > +unlock_free:
> > + spin_unlock_irqrestore(&counter->lock, flags);
> >   out_free:
> >   kfree(buf);
> >   out:
> > Index: mm-2.6.26-rc2-mm1/mm/memcontrol.c
> > =====
```

```
> > --- mm-2.6.26-rc2-mm1.orig/mm/memcontrol.c
> > +@@ mm-2.6.26-rc2-mm1/mm/memcontrol.c
> > @@ -559,7 +559,7 @@ static int mem_cgroup_charge_common(stru
> >   css_get(&memcg->css);
> > }
> >
> > - while (res_counter_charge(&mem->res, PAGE_SIZE)) {
> > + while (res_counter_charge(&mem->res, PAGE_SIZE) == RES_OVER_LIMIT) {
> >   if (!(gfp_mask & __GFP_WAIT))
> >     goto out;
> >
> >
> > Otherwise looks good so far. Need to look at the background reclaim code.
>
```

Thanks,  
-Kame

---

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

---