
Subject: Re: [RFC][only for review] memory controller bacground reclaim [5/5]
Posted by [KAMEZAWA Hiroyuki](#) on Thu, 29 Nov 2007 01:26:09 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Wed, 28 Nov 2007 14:06:22 +0300

Pavel Emelyanov <xemul@openvz.org> wrote:

```
> > + struct {  
> > + wait_queue_head_t waitq;  
> > + struct task_struct *thread;  
> > + } daemon;  
>
```

> Does this HAS to be a struct?

>

No, but for shorter name of member.

(I'd like to add another waitq for avoiding contention in reclaiming,
If I can.)

```
> > + * Background page reclaim daeom for memory controller.
```

```
> > + */
```

```
> > +
```

```
> > +static int mem_cgroup_reclaim_daemon(void *data)
```

```
> > +{
```

```
> > + DEFINE_WAIT(wait);
```

```
> > + struct mem_cgroup *mem = data;
```

```
> > +
```

```
> > + css_get(&mem->css);
```

```
>
```

> Won't this prevent the css from being removed?

>

This refcnt is dropped when this thread dies.

pre_destroy handler handles this at rmdir().

```
> > struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
```

```
> > int ret;
```

```
> > +
```

```
>
```

> This hunk is not needed :)

>

yes :)

```
> > ret = mem_cgroup_force_empty(mem);
```

```
> > if (!ret)
```

```
> > ret = nbytes;
```

```
> > @@ -1188,6 +1244,16 @@
```

```
> >
```

```
> > static struct mem_cgroup init_mem_cgroup;
```

```

> >
> > +static int __init mem_cgroup_reclaim_init(void)
> > +{
> > + init_mem_cgroup.daemon.thread = kthread_run(mem_cgroup_reclaim_daemon,
> > +   &init_mem_cgroup, "memcontd");
> > + if (IS_ERR(init_mem_cgroup.daemon.thread))
> > +   BUG();
> > + return 0;
> > +}
> > +late_initcall(mem_cgroup_reclaim_init);
> > +
> > static struct cgroup_subsys_state *
> > mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
> > {
> > @@ -1212,6 +1278,17 @@
> >   if (alloc_mem_cgroup_per_zone_info(mem, node))
> >     goto free_out;
> >
> > + /* Memory Reclaim Daemon per cgroup */
> > + init_waitqueue_head(&mem->daemon.waitq);
> > + if (mem != &init_mem_cgroup) {
> > +   /* Complicated...but we cannot call kthread create here..*/
> > +   /* init call will later assign kthread */
> > +   mem->daemon.thread = kthread_run(mem_cgroup_reclaim_daemon,
> > +     mem, "memcontd");
> > +   if (IS_ERR(mem->daemon.thread))
> > +     goto free_out;
> >
> > goto free_mem_cgroup_per_zone_info()?
> >
Ah.. there may be some bugs. will fix soon.

```

Thank you for all your comments.

Regards,
-Kame

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