Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by KAMEZAWA Hiroyuki on Wed, 05 Mar 2008 06:53:29 GMT
View Forum Message <> Reply to Message

On Wed, 05 Mar 2008 14:59:05 +0900
Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp> wrote:

> #ifdef CONFIG_CGROUP_MEM_CONT
> +/*
> + * A page_cgroup page is associated with every page descriptor. The
> + * page_cgroup helps us identify information about the cgroup
> + */
> +struct page_cgroup {
> + struct list_head lru; /* per cgroup LRU list */
> + struct page *page;
> + struct mem_cgroup *mem_cgroup;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + struct mm_struct *pc_mm;
> +#endif
> + atomic_t ref_cnt; /* Helpful when pages move b/w */
> +   /* mapped and cached states   */
> + int flags;
> +};
>
As first impression, I don't like to increase size of this...but have no alternative
idea.

> static inline int page_cgroup_locked(struct page *page)
> @@ -664,6 +665,10 @@ retry:
>   pc->flags = PAGE_CGROUP_FLAG_ACTIVE;
>   if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)
>   pc->flags |= PAGE_CGROUP_FLAG_CACHE;
> +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> + atomic_inc(&mm->mm_count);
> + pc->pc_mm = mm;
> +#endif
>
Strongly Nack to this atomic_inc().
What happens when tmpfs pages goes to swap ?

>   if (!page || page_cgroup_assign_new_page_cgroup(page, pc)) {
>   /*
> @@ -673,6 +678,9 @@ retry:

> +int swap_cgroup_charge(struct page *page,

```
> +   struct swap_info_struct *si,
> +   unsigned long offset)
> +{
> + int ret;
> + struct page_cgroup *pc;
> + struct mm_struct *mm;
> + struct swap_cgroup *swap;
> +
> + BUG_ON(!page);
> +
> + /*
> +  * Pages to be swapped out should have been charged by memory cgroup,
> +  * but very rarely, pc would be NULL (pc is not reliable without lock,
> +  * so I should fix here).
> +  * In such cases, we charge the init_mm now.
> +  */
> + pc = page_get_page_cgroup(page);
> + if (WARN_ON(!pc))
> +  mm = &init_mm;
> + else
> +  mm = pc->pc_mm;
> + BUG_ON(!mm);
> +
> + rcu_read_lock();
> + swap = rcu_dereference(mm->swap_cgroup);
> + rcu_read_unlock();
> + BUG_ON(!swap);
```

Is there no race ?

At first look, remembering mm struct is not very good.
Remembering swap controller itself is better.
If you go this direction, how about this way ?

==

```
enum {
#ifdef CONFIG_CGROUP_MEM_CONT
 MEMORY_RESOURCE_CONTROLLER,
#endif
#ifdef CONFIG_CGROUP_SWAP_CONT
 SWAP_CONTROLLER,
#endif
 NR_PAGE_CONTROLLER,
}

struct page_cgroup {
 ......
 void* controlls[NR_PAGE_CONTROLLER];
 ....
```

};
==

Thanks,
-Kame

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

---

Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by Hirokazu Takahashi on Wed, 05 Mar 2008 21:51:53 GMT
View Forum Message <> Reply to Message

Hi,

> > #ifdef CONFIG_CGROUP_MEM_CONT
> > +/*
> > + * A page_cgroup page is associated with every page descriptor. The
> > + * page_cgroup helps us identify information about the cgroup
> > + */
> > +struct page_cgroup {
> > + struct list_head lru; /* per cgroup LRU list */
> > + struct page *page;
> > + struct mem_cgroup *mem_cgroup;
> > +#ifdef CONFIG_CGROUP_SWAP_LIMIT
> > + struct mm_struct *pc_mm;
> > +#endif
> > + atomic_t ref_cnt; /* Helpful when pages move b/w */
> > +   /* mapped and cached states   */
> > + int flags;
> > +};
> >
> As first impression, I don't like to increase size of this...but have no alternative
> idea.

If you really want to make the swap space subsystem and the memory subsystem
work independently each other, you can possibly introduce a new data
structure that binds pages in the swapcache and swap_cgroup.
It would be enough since only a small part of the pages are in the swapcache.

Thanks,

Hirokazu Takahashi.

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

Subject: Re: [RFC/PATCH] cgroup swap subsystem
Posted by Daisuke Nishimura on Thu, 06 Mar 2008 11:45:27 GMT
View Forum Message <> Reply to Message

Hi.

> At first look, remembering mm struct is not very good.
> Remembering swap controller itself is better.

The swap_cgroup when the page(and page_cgroup) is allocated and
the swap_cgroup when the page is going to be swapped out may be
different by swap_cgroup_move_task(), so I think swap_cgroup
to be charged should be determined at the point of swapout.

Instead of pointing mm_struct from page_cgroup, it would be
better to determine the mm_struct which the page to be swapped
out is belongs to by rmap, and charge swap_cgroup of the mm_struct.
In this implementation, I don't need to add new member to page_cgroup.

What do you think ?

Thanks,
Daisuke Nishimura.

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