Subject: Re: [PATCH 2/4] Block I/O tracking
Posted by Hirokazu Takahashi on Tue, 18 Mar 2008 11:34:22 GMT
View Forum Message <> Reply to Message

Hi,

```
> > Hi,
> >
> > This patch implements the bio cgroup on the memory cgroup.
> >
> > Signed-off-by: Hirokazu Takahashi <taka@valinux.co.jp>
> >
> >
> > --- linux-2.6.25-rc5.pagecgroup2/include/linux/memcontrol.h 2008-03-18 12:45:14.000000000
+0900
> > +++ linux-2.6.25-rc5-mm1/include/linux/memcontrol.h 2008-03-18 12:55:59.000000000 +0900
> > @@ -54,6 +54,10 @@ struct page_cgroup {
> >   struct list_head lru;  /* per cgroup LRU list */
> >   struct mem_cgroup *mem_cgroup;
> >  #endif /* CONFIG_CGROUP_MEM_RES_CTLR */
> > +#ifdef  CONFIG_CGROUP_BIO
> > + struct list_head blist;  /* for bio_cgroup page list */
> > + struct bio_cgroup *bio_cgroup;
> > +#endif
>
> Hmm, definition like this
> ==
> enum {
> #ifdef CONFIG_CGROUP_MEM_RES_CTLR
>  MEM_RES_CTLR,
> #endif
> #ifdef CONFIG_CGROURP_BIO
>  BIO_CTLR,
> #endif
>  NR_VM_CTRL,
> };
>
>  void *cgroups[NR_VM_CGROUP];
> ==
> Can save another #ifdefs ?
```

I guess the following code can be possible if you really want to remove
the #ifdefs here. But the both of them look overkill.

```
#ifdef CONFIG_CGROUP_MEM_RES_CTLR
#define MEM_CONTROL \
        struct list_head lru; \
        struct mem_cgroup *mem_cgroup \
#else
#define MEM_CONTROL
#endif

#define BIO_CONTROL \
	struct list_head blist; \
	struct bio_cgroup *bio_cgroup \
#else
#define BIO_CONTROL
#endif

struct page_cgroup {
	MEM_CONTROL;
	BIO_CONTROL;
	struct page *page;
	int ref_cnt;
	int flags;
};
```

> And, blist seems to be just used for force_empty.
> Do you really need this ? no alternative ?

I selected this approach because it was the simplest way for the
first implementation.

I've been also thinking about what you pointed.
If you don't mind taking a long time to remove a bio cgroup, it will be
the easiest way that you can scan all pages to find the pages which
belong to the cgroup and delete them. It may be enough since you may
say it will rarely happen. But it might cause some trouble on machines
with huge memory.

So I'm actually looking for anther way if we could release them lazily.
But I don't think I have to hurry to work on this issue now.

> Thanks,
> -Kame

Thank you,
Hirokazu Takahashi.