Subject: Re: [RFC][PATCH] another swap controller for cgroup
Posted by yamamoto on Wed, 07 May 2008 05:50:44 GMT
View Forum Message <> Reply to Message

hi,

> Hi, Thanks for the patches and your patience. I've just applied your
> patches on top of 2.6.25-mm1 (it had a minor reject, that I've fixed).
> I am building and testing the patches along with KAMEZAWA-San's low
> overhead patches.

thanks.

```
> > +#include <linux/err.h>
> > +#include <linux/cgroup.h>
> > +#include <linux/hugetlb.h>
>
> My powerpc build fails, we need to move hugetlb.h down to the bottom
```

what's the error message?

```
> > +struct swap_cgroup {
> > + struct cgroup_subsys_state scg_css;
>
> Can't we call this just css. Since the structure is swap_cgroup it
> already has the required namespace required to distinguish it from
> other css's. Please see page 4 of "The practice of programming", be
> consistent. The same comment applies to all members below.
```

i don't have the book.
i like this kind of prefixes as it's grep-friendly.

```
> > +#define task_to_css(task) task_subsys_state((task), swap_cgroup_subsys_id)
> > +#define css_to_scg(css) container_of((css), struct swap_cgroup, scg_css)
> > +#define cg_to_css(cg) cgroup_subsys_state((cg), swap_cgroup_subsys_id)
> > +#define cg_to_scg(cg) css_to_scg(cg_to_css(cg))
>
> Aren't static inline better than macros? I would suggest moving to
> them.
```

sounds like a matter of preference.
either ok for me.

```
> > +static struct swap_cgroup *
> > +swap_cgroup_prepare_ptp(struct page *ptp, struct mm_struct *mm)
> > +{
> > + struct swap_cgroup *scg = ptp->ptp_swap_cgroup;
> > +
```

>
> Is this routine safe w.r.t. concurrent operations, modifications to
> ptp_swap_cgroup?

it's always accessed with the page table locked.

```
> > + BUG_ON(mm == NULL);
> > + BUG_ON(mm->swap_cgroup == NULL);
> > + if (scg == NULL) {
> > +  /*
> > +   * see swap_cgroup_attach.
> > +   */
> > +  smp_rmb();
> > +  scg = mm->swap_cgroup;
>
```

> With the mm->owner patches, we need not maintain a separate
> mm->swap_cgroup.

i don't think the mm->owner patch, at least with the current form,
can replace it.

```
> > + /*
> > +  * swap_cgroup_attach is in progress.
> > +  */
> > +
> > + res_counter_charge_force(&newscg->scg_counter, PAGE_CACHE_SIZE);
>
```

> So, we force the counter to go over limit?

yes.

as newscg != oldscg here means the task is being moved between cgroups,
this instance of res_counter_charge_force should not matter much.

```
> > +static int
> > +swap_cgroup_write_u64(struct cgroup *cg, struct cftype *cft, u64 val)
> > +{
> > + struct res_counter *counter = &cg_to_scg(cg)->scg_counter;
> > + unsigned long flags;
> > +
> > + /* XXX res_counter_write_u64 */
> > + BUG_ON(cft->private != RES_LIMIT);
> > + spin_lock_irqsave(&counter->lock, flags);
> > + counter->limit = val;
> > + spin_unlock_irqrestore(&counter->lock, flags);
> > + return 0;
> > +}
> > +
```

>
> We need to write actual numbers here? Can't we keep the write
> interface consistent with the memory controller?

i'm not sure what you mean here. can you explain a bit more?
do you mean K, M, etc?

```
> > +static void
> > +swap_cgroup_destroy(struct cgroup_subsys *ss, struct cgroup *cg)
> > +{
> > + struct swap_cgroup *oldscg = cg_to_scg(cg);
> > + struct swap_cgroup *newscg;
> > + struct list_head *pos;
> > + struct list_head *next;
> > +
> > + /*
> > +  * move our anonymous objects to init_mm's group.
> > +  */
```

>
> Is this good design, should be allow a swap_cgroup to be destroyed,
> even though pages are allocated to it?

first of all, i'm not quite happy with this design. :)
having said that, what else can we do?
i tend to think that trying to swap-in these pages is too much effort
for little benefit.

> Is moving to init_mm (root
> cgroup) a good idea? Ideally with support for hierarchies, if we ever
> do move things, it should be to the parent cgroup.

i chose init_mm because there seemed to be no consensus about
cgroup hierarchy semantics.

```
> > +  info->swap_cgroup = newscg;
> > +  res_counter_uncharge(&oldscg->scg_counter, bytes);
> > +  res_counter_charge_force(&newscg->scg_counter, bytes);
```

>
> I don't like the excessive use of res_counter_charge_force(), it seems
> like we might end up bypassing the controller all together. I would
> rather fail the destroy operation if the charge fails.

```
> > + bytes = swslots * PAGE_CACHE_SIZE;
> > + res_counter_uncharge(&oldscg->scg_counter, bytes);
> > + /*
> > +  * XXX ignore newscg's limit because cgroup ->attach method can't fail.
> > +  */
> > + res_counter_charge_force(&newscg->scg_counter, bytes);
```

>
> But in the future, we could plan on making attach fail (I have a
> requirement for it). Again, I don't like the _force operation

allowing these operations fail implies to have code to back out
partial operations. i'm afraid that it will be too complex.

```
> > +static void
> > +swap_cgroup_attach_mm(struct mm_struct *mm, struct swap_cgroup *oldscg,
> > +    struct swap_cgroup *newscg)
```

>
> We need comments about the function, why do we attach an mm?

instead of a task, you mean?
because we count the number of ptes which points to swap
and ptes belong to an mm, not a task.

YAMAMOTO Takashi