Subject: Re: [ckrm-tech] [RFC][PATCH][2/4] Add RSS accounting and control
Posted by Andrew Morton on Mon, 19 Feb 2007 11:01:41 GMT
View Forum Message <> Reply to Message

On Mon, 19 Feb 2007 16:07:44 +0530 Balbir Singh <balbir@in.ibm.com> wrote:

> >> +void memctlr_mm_free(struct mm_struct *mm)
> >> +{
> >> + kfree(mm->counter);
> >> +}
> >> +
> >> +static inline void memctlr_mm_assign_container_direct(struct mm_struct *mm,
> >> + struct container *cont)
> >> +{
> >> + write_lock(&mm->container_lock);
> >> + mm->container = cont;
> >> + write_unlock(&mm->container_lock);
> >> +}
> >
> > More weird locking here.
> >
>
> The container field of the mm_struct is protected by a read write spin lock.

That doesn't mean anything to me.

What would go wrong if the above locking was simply removed? And how does
the locking prevent that fault?

> >> +void memctlr_mm_assign_container(struct mm_struct *mm, struct task_struct *p)
> >> +{
> >> + struct container *cont = task_container(p, &memctlr_subsys);
> >> + struct memctlr *mem = memctlr_from_cont(cont);
> >> +
> >> + BUG_ON(!mem);
> >> + write_lock(&mm->container_lock);
> >> + mm->container = cont;
> >> + write_unlock(&mm->container_lock);
> >> +}
> >
> > And here.
> >
>
> Ditto.

ditto ;)

> >

> >> +/*
> >> + * Update the rss usage counters for the mm_struct and the container it belongs
> >> + * to. We do not fail rss for pages shared during fork (see copy_one_pte()).
> >> + */
> >> +int memctlr_update_rss(struct mm_struct *mm, int count, bool check)
> >> +{
> >> + int ret = 1;
> >> + struct container *cont;
> >> + long usage, limit;
> >> + struct memctlr *mem;
> >> +
> >> + read_lock(&mm->container_lock);
> >> + cont = mm->container;
> >> + read_unlock(&mm->container_lock);
> >> +
> >> + if (!cont)
> >> + goto done;
> >
> > And here. I mean, if there was a reason for taking the lock around that
> > read, then testing `cont' outside the lock just invalidated that reason.
> >
>
> We took a consistent snapshot of cont. It cannot change outside the lock,
> we check the value outside. I am sure I missed something.

If it cannot change outside the lock then we don't need to take the lock!

>
> MEMCTLR_DONT_CHECK_LIMIT exists for the following reasons
>
> 1. Pages are shared during fork, fork() is not failed at that point
>   since the pages are shared anyway, we allow the RSS limit to be
>   exceeded.
> 2. When ZERO_PAGE is added, we don't check for limits (zeromap_pte_range).
> 3. On reducing RSS (passing -1 as the value)

OK, that might make a nice comment somewhere (if it's not already there).
