Subject: Re: [PATCH 6/6] containers: BeanCounters over generic process containers
Posted by Paul Menage on Sun, 24 Dec 2006 11:32:25 GMT
View Forum Message <> Reply to Message

On 12/23/06, Herbert Poetzl <herbert@13thfloor.at> wrote:
> On Fri, Dec 22, 2006 at 06:14:48AM -0800, Paul Menage wrote:
> > This patch implements the BeanCounter resource control abstraction
> > over generic process containers. It contains the beancounter core
> > code, plus the numfiles resource counter. It doesn't currently contain
> > any of the memory tracking code or the code for switching beancounter
> > context in interrupts.
>
> I don't like it, it looks bloated and probably
> adds plenty of overhead (similar to the OVZ
> implementation where this seems to be taken from)

Yes - perhaps I should have been clearer in the patch description. It's basically code taken from the OpenVZ bean counters patches that have been posted recently, but with the filesystem and process tracking code ripped out (since it's implemented over the generic containers).

The main point of this patch is to demonstrate that UBC can be implemented effectively over generic containers, rather than to be a proposal in favour of UBC versus any of the other potential resource control mechanisms.

Most of your comments are about code taken pretty much directly from the UBC patches, so I won't address them.

>
> > +int bc_file_charge(struct file *file)
> > +{
> > +	int sev;
> > +	struct beancounter *bc;
> > +
> > +	task_lock(current);
>
> why do we lock current? it won't go away that
> easily, and for switching the bc, it might be
> better to use RCU or a separate lock, no?
>

The locking model (taken originally from the Cpusets code) in generic containers is that while you can use RCU to guarantee that a pointer read from current->container remains valid until you exit the RCU critical section, if you want to make consistent changes to data

referenced from a task P's container, you need to hold either
P->alloc_lock or one of the two container mutexes (manage_mutex and/or
callback_mutex).

In this particular case (sorry, not on the VPN right now to be able to
figure out the potential code changes) the fact that the call to
css_get_current() uses atomic operations (currently a spinlock, but I
suspect I could optimize it to be a cmpxchg) could mean that we can
skip the task_lock(), at the cost of occasionally accounting a file to
the container that the task had just left.

Paul

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers