Subject: Re: [ckrm-tech] [RFC][PATCH 2/7] UBC: core (structures, API)
Posted by Matt Helsley on Sat, 19 Aug 2006 02:38:36 GMT
View Forum Message <> Reply to Message

On Fri, 2006-08-18 at 15:36 +0400, Kirill Korotaev wrote:
> Matt Helsley wrote:

<snip>

> >>+ spin_unlock_irqrestore(&ub_hash_lock, flags);
> >>+ return;
> >>+ }
> >>+
> >>+ verify_held(ub);
> >>+ hlist_del(&ub->hash);
> >>+ spin_unlock_irqrestore(&ub_hash_lock, flags);
> >>+
> >>+ kmem_cache_free(ub_cachep, ub);
> >>+
> >>+ ub = parent;
> >>+ if (ub != NULL)
> >>+ goto again;
> >>+
> >
> > Couldn't this be replaced by a do { } while (ub != NULL); loop?
> this is ugly from indentation POV. also restarts are frequently used everywhere...

Then perhaps the body could be made into a small function or set of
functions.

I know the retry pattern is common. Though, as I remember it the control
flow was much more complex when goto was used for retry. Also, I seem to
recall do {} while () has favorable properties that goto lacks when it
comes to compiler optimization.

<snip>

> >>+int charge_beancounter(struct user_beancounter *ub,
> >>+ int resource, unsigned long val, enum severity strict)
> >>+{
> >>+ int retval;
> >>+ struct user_beancounter *p, *q;
> >>+ unsigned long flags;
> >>+
> >>+ retval = -EINVAL;
> >>+ BUG_ON(val > UB_MAXVALUE);
> >>+
> >>+ local_irq_save(flags);

> >
> >
> > <factor>
> >
> >>+ for (p = ub; p != NULL; p = p->parent) {
> >
> >
> > Seems rather expensive to walk up the tree for every charge. Especially
> > if the administrator wants a fine degree of resource control and makes a
> > tall tree. This would be a problem especially when it comes to resources
> > that require frequent and fast allocation.
> in heirarchical accounting you always have to update all the nodes :/
> with flat UBC this doesn't introduce significant overhead.

Except that you eventually have to lock ub0. Seems that the cache line
for that spinlock could bounce quite a bit in such a hot path.

Chandra, doesn't Resource Groups avoid walking more than 1 level up the
hierarchy in the "charge" paths?

> >>+ spin_lock(&p->ub_lock);
> >>+ retval = __charge_beancounter_locked(p, resource, val, strict);
> >>+ spin_unlock(&p->ub_lock);
> >>+ if (retval)
> >>+   goto unroll;
> >
> >
> > This can be factored by passing a flag that breaks the loop on an error:
> >
> >   if (retval && do_break_err)
> >   return retval;
> how about uncharge here?
> didn't get your idea, sorry...

 The only structural difference between this loop and another you have
is the "break" here. I was saying that you could pass a parameter into
the factored portion that tells it to return early if there is an error.

Cheers,
 -Matt Helsley