Subject: Re: [PATCH 6/6] containers: BeanCounters over generic process containers
Posted by Herbert Poetzl on Sat, 23 Dec 2006 19:49:55 GMT
View Forum Message <> Reply to Message

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)
here are some comments/questions:

> Currently all the beancounters resource counters are lumped into a
> single hierarchy; ideally it would be possible for each resource
> counter to be a separate container subsystem, allowing them to be
> connected to different hierarchies.
>
> +static inline void bc_uncharge(struct beancounter *bc, int res_id,
> + unsigned long val)
> +{
> + unsigned long flags;
> +
> + spin_lock_irqsave(&bc->bc_lock, flags);
> + bc_uncharge_locked(bc, res_id, val);
> + spin_unlock_irqrestore(&bc->bc_lock, flags);

why use a spinlock, when we could use atomic
counters?

> +int bc_charge_locked(struct beancounter *bc, int res, unsigned long val,
> + int strict, unsigned long flags)
> +{
> + struct bc_resource_parm *parm;
> + unsigned long new_held;
> +
> + BUG_ON(val > BC_MAXVALUE);
> +
> + parm = &bc->bc_parms[res];
> + new_held = parm->held + val;
> +
> + switch (strict) {
> + case BC_LIMIT:
> + if (new_held > parm->limit)

```
> +   break;
> +  /* fallthrough */
> + case BC_BARRIER:
> +  if (new_held > parm->barrier) {
> +   if (strict == BC_BARRIER)
> +    break;
> +   if (parm->held < parm->barrier &&
> +     bc_resources[res]->bcr_barrier_hit)
> +    bc_resources[res]->bcr_barrier_hit(bc);
> +  }
```

why do barrier checks with every accounting?
there are probably a few cases where the
checks could be independant from the accounting

```
> +  /* fallthrough */
> + case BC_FORCE:
> +  parm->held = new_held;
> +  bc_adjust_maxheld(parm);
```

in what cases do we want to cross the barrier?

```
> +  return 0;
> + default:
> +  BUG();
> + }
> +
> + if (bc_resources[res]->bcr_limit_hit)
> +  return bc_resources[res]->bcr_limit_hit(bc, val, flags);
> +
> + parm->failcnt++;
> + return -ENOMEM;

> +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?

```
> + bc = task_bc(current);
> + css_get_current(&bc->css);
> + task_unlock(current);
> +
```

```
> + sev = (capable(CAP_SYS_ADMIN) ? BC_LIMIT : BC_BARRIER);
> +
> + if (bc_charge(bc, BC_NUMFILES, 1, sev)) {
> +  css_put(&bc->css);
> +  return -EMFILE;
> + }
> +
> + file->f_bc = bc;
> + return 0;
> +}
```

also note that certain limits are much more
complicated than the (very simple) file limits
and the code will be called at higher frequency

how to handle requests like:
 try to get as 64 files or as many as available
 whatever is smaller

happy xmas,
Herbert

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