Subject: Re: [ckrm-tech] [PATCH 5/13] BC: user interface (syscalls)
Posted by Pavel Emelianov on Wed, 06 Sep 2006 10:42:48 GMT
View Forum Message <> Reply to Message

```
Balbir Singh wrote:
> Pavel Emelianov wrote:
>> Balbir Singh wrote:
>>>> +
>>>> +asmlinkage long sys_set_bcid(bcid_t id)
>>>> +{
>>>> +    int error;
>>>> +    struct beancounter *bc;
>>>> +    struct task_beancounter *task_bc;
>>>> +
>>>> +    task_bc = &current->task_bc;
>>> I was playing around with the bc patches and found that to make
>>> use of bc's, I had to actually call set_bcid() and then exec() a
>>> task/shell so that the id would stick around. Would you consider
>> That sounds very strange as sys_set_bcid() actually changes current's
>> exec_bc.
>> One note is about mm's bc - mm obtains new bc only after fork or exec -
>> that's
>> true. But kmemsize starts charging right after the sys_set_bcid.
>
> I was playing around only with kmemsize. I think the reason for my
> observation
> is this
>
> bash --> (my utility) --> set_bcid()
>
> Since bash spawns my utility in a separate process, it creates and
> assigns
> a bean counter to it and then my utility exits. Unless it
> spawns/exec()'s a
> new shell, the beancounter is freed when the task exits (my utility).
Well, beancounter is not "inherited" by parent task :)
After setting bcid you need to spawn/exec a new shell.
But seeting limits and getting stats is possible from the old shell
as well as from the new one.
>
>>> changing sys_set_bcid to sys_set_task_bcid() or adding a new
>>> system call sys_set_task_bcid()? We could pass the pid that we
>>> intend to associate with the new id. This also means we'll need
>>> locking around to protect task->task_bc.
>>
>
>
```