Subject: Re: [ckrm-tech] [PATCH] BC: resource beancounters (v4) (added user memory)
Posted by Pavel Emelianov on Mon, 11 Sep 2006 07:02:06 GMT
View Forum Message <> Reply to Message

Chandra Seetharaman wrote:
> On Fri, 2006-09-08 at 11:22 +0400, Pavel Emelianov wrote:
>
>> Chandra Seetharaman wrote:
>>
>> [snip]
>>
>>>>>> The question is - whether web server is multithreaded or not...
>>>>>> If it is not - then no problem here, you can change current
>>>>>> context and new resources will be charged accordingly.
>>>>>>
>>>>>> And current BC code is _able_ to handle it with _minor_ changes.
>>>>>> (One just need to save bc not on mm struct, but rather on vma struct
>>>>>> and change mm->bc on set_bc_id()).
>>>>>>
>>>>>> However, no one (can some one from CKRM team please?) explained so far
>>>>>> what to do with threads. Consider the following example.
>>>>>>
>>>>>> 1. Threaded web server spawns a child to serve a client.
>>>>>> 2. child thread touches some pages and they are charged to child BC
>>>>>>   (which differs from parent's one)
>>>>>> 3. child exits, but since its mm is shared with parent, these pages
>>>>>>   stay mapped and charged to child BC.
>>>>>>
>>>>>> So the question is: what to do with these pages?
>>>>>> - should we recharge them to another BC?
>>>>>> - leave them charged?
>>>>>>
>>>>>>
>>>>>>
>>>>> Leave them charged. It will be charged to the appropriate UBC when they
>>>>> touch it again.
>>>>>
>>>>>
>>>>>
>>>> Do you mean that page must be re-charged each time someone touches it?
>>>>
>>>>
>>> What I meant is that to leave them charged, and if when they are
>>> ummapped and mapped later, charge it to the appropriate BC.
>>>
>>>
>> In this case multithreaded apache that tries to serve each domain in

>> separate BC will fill the memory with BC-s, held by pages allocated
>> and mapped in threads.
>>
>
> I do not understand how the memory will be filled with BCs. Can you
> explain, please.
>
Sure. At the beginning I have one task with one BC. Then
1. A thread is spawned and new BC is created;
2. New thread touches a new page (e.g. maps a new file) which is charged to new BC
   (and this means that this BC's must stay in memory till page is uncharged);
3. Thread exits after serving the request, but since it's mm is shared with parent
   all the touched pages stay resident and, thus, the new BC is still pinned in memory.
Steps 1-3 are done multiple times for new pages (new files).
Remember that we're discussing the case when pages are not recharged.