Subject: Re: [ckrm-tech] [PATCH] BC: resource beancounters (v4) (added user memory)
Posted by Pavel Emelianov on Tue, 12 Sep 2006 11:06:35 GMT
View Forum Message <> Reply to Message

Srivatsa Vaddagiri wrote:
> On Tue, Sep 12, 2006 at 02:24:25PM +0400, Pavel Emelianov wrote:
>
>> Srivatsa Vaddagiri wrote:
>>
>>> On Mon, Sep 11, 2006 at 11:02:06AM +0400, Pavel Emelianov wrote:
>>>
>>>
>>>> Sure. At the beginning I have one task with one BC. Then
>>>> 1. A thread is spawned and new BC is created;
>>>>
>>>>
>>> Why do we have to create a BC for every new thread? A new BC is needed
>>> for every new service level instead IMO. And typically there wont be
>>> unlimited service levels.
>>>
>>>
>> That's the scenario we started from - each domain is served in a separate
>> BC with *threaded* Apache.
>>
>
> Sure ..but you can still meet that requirement by creating fixed set of
> BCs (for each domain) and let each new thread be associated with a
> corresponding BC (w/o requiring to create BC for every new thread),
> depending on which domain's request it is serving?
>
Hmmm... Beancounters can provide this after trivial changes.
We may schedule them in current set of "pending" features
(http://wiki.openvz.org/UBC_discussion)

But this can create a kind of DoS within an application:
  A thread continuously touches new and new pages to it's BC and
these pages are get touched by other threads also. Sooner or later
this BC will hit it's limit and reclaiming this set of pages would affect
all the other threads.

Also such accounting reveals you NOTHING about real memory usage.
E.g. 100Mb charged for one BC can mean "this BC ate 100Mb of
memory" as well as "this BC uses one page really, but all the others
are just used by other threads" and anything between these two
corner cases.

Well. We've digressed from our main thread - discussing (dis)advantages

of current BC implemenation.
>
>>>
>>>
>>>> 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.
>>>>