Subject: Re: [PATCH v3 13/28] slub: create duplicate cache
Posted by Suleiman Souhlal on Tue, 29 May 2012 20:57:41 GMT
View Forum Message <> Reply to Message

Hi Christoph,

On Tue, May 29, 2012 at 1:21 PM, Christoph Lameter <cl@linux.com> wrote:
> On Wed, 30 May 2012, Glauber Costa wrote:
>
>> Well, I'd have to dive in the code a bit more, but that the impression that
>> the documentation gives me, by saying:
>>
>> "Cpusets constrain the CPU and Memory placement of tasks to only
>> the resources within a task's current cpuset."
>>
>> is that you can't allocate from a node outside that set. Is this correct?
>
> Basically yes but there are exceptions (like slab queues etc). Look at the
> hardwall stuff too that allows more exceptions for kernel allocations to
> use memory from other nodes.
>
>> So extrapolating this to memcg, the situation is as follows:
>>
>> * You can't use more memory than what you are assigned to.
>> * In order to do that, you need to account the memory you are using
>> * and to account the memory you are using, all objects in the page
>>   must belong to you.
>
> Cpusets work at the page boundary and they do not have the requirement you
> are mentioning of all objects in the page having to belong to a certain
> cpusets. Let that go and things become much easier.
>
>> With a predictable enough workload, this is a recipe for working around the
>> very protection we need to establish: one can DoS a physical box full of
>> containers, by always allocating in someone else's pages, and pinning kernel
>> memory down. Never releasing it, so the shrinkers are useless.
>
> Sure you can construct hyperthetical cases like that. But then that is
> true already of other container like logic in the kernel already.
>
>> So I still believe that if a page is allocated to a cgroup, all the objects in
>> there belong to it - unless of course the sharing actually means something -
>> and identifying this is just too complicated.
>
> We have never worked container like logic like that in the kernel due to
> the complicated logic you would have to put in. The requirement that all
> objects in a page come from the same container is not necessary. If you
> drop this notion then things become very easy and the patches will become

> simple.

Back when we (Google) started using cpusets for memory isolation (fake NUMA), we found that there was a significant isolation breakage coming from slab pages belonging to one cpuset being used by other cpusets, which caused us problems: It was very easy for one job to cause slab growth in another container, which would cause it to OOM, despite being well-behaved.

Because of this, we had to add logic to prevent that from happening (by making sure we only allocate objects from pages coming from our allowed nodes).

Now that we're switching to doing containers with memcg, I think this is a hard requirement, for us. :-(

-- Suleiman
