Subject: Re: [PATCH v4 14/14] Add documentation about the kmem controller
Posted by Glauber Costa on Fri, 12 Oct 2012 07:53:23 GMT
View Forum Message <> Reply to Message

On 10/11/2012 06:35 PM, Michal Hocko wrote:
> On Mon 08-10-12 14:06:20, Glauber Costa wrote:
>> Signed-off-by: Glauber Costa <glommer@parallels.com>
>> ---
>> Documentation/cgroups/memory.txt | 55 ++-
>> 1 file changed, 54 insertions(+), 1 deletion(-)
>>
>> diff --git a/Documentation/cgroups/memory.txt b/Documentation/cgroups/memory.txt
>> index c07f7b4..9b08548 100644
>> --- a/Documentation/cgroups/memory.txt
>> +++ b/Documentation/cgroups/memory.txt
>> @@ -71,6 +71,11 @@ Brief summary of control files.
>>   memory.oom_control   # set/show oom controls.
>>   memory.numa_stat   # show the number of memory usage per numa node
>>
>> + memory.kmem.limit_in_bytes   # set/show hard limit for kernel memory
>> + memory.kmem.usage_in_bytes   # show current kernel memory allocation
>> + memory.kmem.failcnt   # show the number of kernel memory usage hits limits
>> + memory.kmem.max_usage_in_bytes # show max kernel memory usage recorded
>> +
>>   memory.kmem.tcp.limit_in_bytes # set/show hard limit for tcp buf memory
>>   memory.kmem.tcp.usage_in_bytes # show current tcp buf memory allocation
>>   memory.kmem.tcp.failcnt   # show the number of tcp buf memory usage hits limits
>> @@ -268,20 +273,62 @@ the amount of kernel memory used by the system. Kernel memory
is fundamentally
>> different than user memory, since it can't be swapped out, which makes it
>> possible to DoS the system by consuming too much of this precious resource.
>>
>> +Kernel memory won't be accounted at all until it is limited. This allows for
>
> until limit on a group is set.
>
ok.

>> +existing setups to continue working without disruption. Note that it is
>> +possible to account it without an effective limit by setting the limits
>> +to a very high number (like RESOURCE_MAX -1page).
>
> I have brought that up in an earlier patch already. Why not just do echo
> -1 (which translates to RESOURCE_MAX internally) and be done with that.
> RESOURCE_MAX-1 sounds quite inconvenient.
>

For the case that you are limited already, and then want to unlimit,

keeping the accounting, yes, it makes sense.

>> The limit cannot be set
>> +if the cgroup have children, or if there are already tasks in the cgroup.
>
> I would start by stating that if children are accounted automatically if
> their parent is accounted already and there is no need to set a limit to
> enforce that. In fact the limit cannot be set if ....
>

ok.

>> +
>> +After a controller is first limited, it will be kept being accounted until it
>
> group is limited not the controller.
>

true, thanks.

>> +
>> Kernel memory limits are not imposed for the root cgroup. Usage for the root
>> -cgroup may or may not be accounted.
>> +cgroup may or may not be accounted. The memory used is accumulated into
>> +memory.kmem.usage_in_bytes, or in a separate counter when it makes sense.
>
> Which separate counter? Is this about tcp kmem?
>

So far, yes, this is the only case that makes sense, and the fewer the
better. In any case it exists, and I wanted to be generic.

>> +The main "kmem" counter is fed into the main counter, so kmem charges will
>> +also be visible from the user counter.
>>
>> Currently no soft limit is implemented for kernel memory. It is future work
>> to trigger slab reclaim when those limits are reached.
>>
>> 2.7.1 Current Kernel Memory resources accounted
>>
>> +* stack pages: every process consumes some stack pages. By accounting into
>> +kernel memory, we prevent new processes from being created when the kernel
>> +memory usage is too high.
>> +
>> * sockets memory pressure: some sockets protocols have memory pressure
>> thresholds. The Memory Controller allows them to be controlled individually
>> per cgroup, instead of globally.

>>
>> * tcp memory pressure: sockets memory pressure for the tcp protocol.
>>
>> +2.7.3 Common use cases
>> +
>> +Because the "kmem" counter is fed to the main user counter, kernel memory can
>> +never be limited completely independently of user memory. Say "U" is the user
>> +limit, and "K" the kernel limit. There are three possible ways limits can be
>> +set:
>> +
>> +   U != 0, K = 0:
>
> K is not 0 it is unaccounted (disabled)
>
>> +   This is the standard memcg limitation mechanism already present before kmem
>> +   accounting. Kernel memory is completely ignored.
>> +
>> +   U,K != 0, K < U:
>
> I would keep K < U
>> +   Kernel memory is effectively set as a percentage of the user memory. This
>
> not a percentage it is subset of the user memory
>
Well, this is semantics. I can change, but for me it makes a lot of
sense to think of it in terms of a percentage, because it is easy to
administer. You don't actually write a percentage, which I tried to
clarify by using the term "effective set as a percentage".

>> +   setup is useful in deployments where the total amount of memory per-cgroup
>> +   is overcommited. Overcommiting kernel memory limits is definitely not
>> +   recommended, since the box can still run out of non-reclaimable memory.
>> +   In this case, the admin could set up K so that the sum of all groups is
>> +   never greater than the total memory, and freely set U at the cost of his
>> +   QoS.
>> +
>> +   U,K != 0, K >= U:
>> +   Since kmem charges will also be fed to the user counter, this setup gives
>> +   the admin a unified view of memory. Reclaim will be triggered for the cgroup
>> +   for both kinds of memory.
>
> This is also useful for tracking kernel memory allocation.
>
ok.