Subject: Re: [PATCH v9 1/9] Basic kernel memory functionality for the Memory Controller
Posted by Glauber Costa on Thu, 15 Dec 2011 12:29:18 GMT
View Forum Message <> Reply to Message

On 12/14/2011 09:04 PM, Michal Hocko wrote:
> [Now with the current patch version, I hope]
>
> On Mon 12-12-11 11:47:01, Glauber Costa wrote:
>> This patch lays down the foundation for the kernel memory component
>> of the Memory Controller.
>>
>> As of today, I am only laying down the following files:
>>
>> * memory.independent_kmem_limit
>
> Maybe has been already discussed but the name is rather awkward and it
> would deserve more clarification. It is independent in the way that it
> doesn't add up to the standard (user) allocations or it enables/disables
> accounting?

If turned on, it doesn't add up to the user allocations.
As for the name, this is marked experimental, so I don't think anyone
will be relying on it for a while. We can change it, if you have a
better suggestion.

>> * memory.kmem.limit_in_bytes (currently ignored)
>
> What happens if we reach the limit? Are all kernel allocations
> considered or only selected caches? How do I find out which are those?
>
> AFAIU you have implemented it for network buffers at this stage but I
> guess that dentries will follow...

Further allocations should fail.

About other caches, tcp is a bit different because we are concerned with
conditions that applies after the allocation already took place. It is
not clear to me if we will treat the other caches as a single entity, or
separate them.

>> * memory.kmem.usage_in_bytes (always zero)
>>
>> Signed-off-by: Glauber Costa<glommer@parallels.com>
>> CC: Kirill A. Shutemov<kirill@shutemov.name>
>> CC: Paul Menage<paul@paulmenage.org>
>> CC: Greg Thelen<gthelen@google.com>
>> CC: Johannes Weiner<jweiner@redhat.com>

>> CC: Michal Hocko<mhocko@suse.cz>
>> ---
>> Documentation/cgroups/memory.txt | 40 +++++++++++++++-
>> init/Kconfig | 11 ++++
>> mm/memcontrol.c | 105 +++--
>> 3 files changed, 149 insertions(+), 7 deletions(-)
>>
>> diff --git a/Documentation/cgroups/memory.txt b/Documentation/cgroups/memory.txt
>> index cc0ebc5..f245324 100644
>> --- a/Documentation/cgroups/memory.txt
>> +++ b/Documentation/cgroups/memory.txt
>> @@ -44,8 +44,9 @@ Features:
>>   - oom-killer disable knob and oom-notifier
>>   - Root cgroup has no limit controls.
>>
>> - Kernel memory and Hugepages are not under control yet. We just manage
>> - pages on LRU. To add more controls, we have to take care of performance.
>> + Hugepages is not under control yet. We just manage pages on LRU. To add more
>
> Hugepages are not
> Anyway this sounds outdated as we track both THP and hugetlb, right?
>
>> + controls, we have to take care of performance. Kernel memory support is work
>> + in progress, and the current version provides basically functionality.
>
> s/basically/basic/
>
>>
>> Brief summary of control files.
>>
>> @@ -56,8 +57,11 @@ Brief summary of control files.
>>   (See 5.5 for details)
>>   memory.memsw.usage_in_bytes # show current res_counter usage for memory+Swap
>>   (See 5.5 for details)
>> + memory.kmem.usage_in_bytes # show current res_counter usage for kmem only.
>> +   (See 2.7 for details)
>>   memory.limit_in_bytes   # set/show limit of memory usage
>>   memory.memsw.limit_in_bytes # set/show limit of memory+Swap usage
>> + memory.kmem.limit_in_bytes # if allowed, set/show limit of kernel memory
>>   memory.failcnt   # show the number of memory usage hits limits
>>   memory.memsw.failcnt   # show the number of memory+Swap hits limits
>>   memory.max_usage_in_bytes # show max memory usage recorded
>> @@ -72,6 +76,9 @@ 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.independent_kmem_limit # select whether or not kernel memory limits are
>> +   independent of user limits

>> +
>
> It is not clear what happens in enabled/disabled cases. Let's say they
> are not independent. Does it form a single limit with user charges or it
> toggles kmem charging on/off.
>
>>   1. History
>>
>>   The memory controller has a long history. A request for comments for the memory
>> @@ -255,6 +262,35 @@ When oom event notifier is registered, event will be delivered.
>>   per-zone-per-cgroup LRU (cgroup's private LRU) is just guarded by
>>   zone->lru_lock, it has no lock of its own.
>>
>> +2.7 Kernel Memory Extension (CONFIG_CGROUP_MEM_RES_CTLR_KMEM)
>> +
>> +With the Kernel memory extension, the Memory Controller is able to limit
>> +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.
>> +
>> +Some kernel memory resources may be accounted and limited separately from the
>> +main "kmem" resource. For instance, a slab cache that is considered important
>> +enough to be limited separately may have its own knobs.
>
> How do you tell which are those that are accounted to the "main kmem"?

Besides being in this list, they should have they own files, like tcp.
>
>> +
>> +Kernel memory limits are not imposed for the root cgroup. Usage for the root
>> +cgroup may or may not be accounted.
>> +
>> +Memory limits as specified by the standard Memory Controller may or may not
>> +take kernel memory into consideration. This is achieved through the file
>> +memory.independent_kmem_limit. A Value different than 0 will allow for kernel
>> +memory to be controlled separately.
>
> Separately from user space allocations, right?
Yes.
> What happens if we reach the limit in both cases?
For kernel memory, further allocations should fail.

>
>> @@ -344,9 +353,14 @@ enum charge_type {
>>   };
>>
>>   /* for encoding cft->private value on file */
>> -#define _MEM   (0)

>> -#define _MEMSWAP (1)
>> -#define _OOM_TYPE (2)
>> +
>> +enum mem_type {
>> + _MEM = 0,
>> + _MEMSWAP,
>> + _OOM_TYPE,
>> + _KMEM,
>> +};
>> +
>
> Probably in a separate (cleanup) patch?
>
>>   #define MEMFILE_PRIVATE(x, val) (((x)<< 16) | (val))
>>   #define MEMFILE_TYPE(val) (((val)>> 16)& 0xffff)
>>   #define MEMFILE_ATTR(val) ((val)& 0xffff)
>> @@ -3848,10 +3862,17 @@ static inline u64 mem_cgroup_usage(struct mem_cgroup
*memcg, bool swap)
>>   u64 val;
>>
>>   if (!mem_cgroup_is_root(memcg)) {
>> + val = 0;
>> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
>> + if (!memcg->kmem_independent_accounting)
>> +   val = res_counter_read_u64(&memcg->kmem, RES_USAGE);
>> +#endif
>>   if (!swap)
>> -   return res_counter_read_u64(&memcg->res, RES_USAGE);
>> +   val += res_counter_read_u64(&memcg->res, RES_USAGE);
>>   else
>> -   return res_counter_read_u64(&memcg->memsw, RES_USAGE);
>> +   val += res_counter_read_u64(&memcg->memsw, RES_USAGE);
>> +
>> + return val;
>>   }
>
> So you report kmem+user but we do not consider kmem during charge so one
> can easily end up with usage_in_bytes over limit but no reclaim is going
> on. Not good, I would say.
>
> OK, so to sum it up. The biggest problem I see is the (non)independent
> accounting. We simply cannot mix user+kernel limits otherwise we would
> see issues (like kernel resource hog would force memcg-oom and innocent
> members would die because their rss is much bigger).
> It is also not clear to me what should happen when we hit the kmem
> limit. I guess it will be kmem cache dependent.

So right now, tcp is completely independent, since it is not accounted

to kmem. In summary, we still never do non-independent accounting. When we start doing it for the other caches, We will have to add a test at charge time as well.

We still need to keep it separate though, in case the independent flag is turned on/off

---