Hi,

This is the first part of the kernel memory controller for memcg. It has been discussed many times, and I consider this stable enough to be on tree. A follow up to this series are the patches to also track slab memory. They are not included here because I believe we could benefit from merging them separately for better testing coverage. If there are any issues preventing this to be merged, let me know. I'll be happy to address them.

*v5:    - changed charged order, kmem charged first.
       - minor nits and comments merged.

*v4:    - kmem_accounted can no longer become unlimited
       - kmem_accounted can no longer become limited, if group has children.
       - documentation moved to this patchset
       - more style changes
       - css_get in charge path to ensure task won't move during charge

*v3:
- Changed function names to match memcg's
- avoid doing get/put in charge/uncharge path
- revert back to keeping the account enabled after it is first activated

A (throwaway) git tree with them is placed at:

git://git.kernel.org/pub/scm/linux/kernel/git/glommer/memcg. git kmemcg-stack

A general explanation of what this is all about follows:

The kernel memory limitation mechanism for memcg concerns itself with disallowing potentially non-reclaimable allocations to happen in exaggerate quantities by a particular set of processes (cgroup). Those allocations could create pressure that affects the behavior of a different and unrelated set of processes.

One example of such problematic pressure that can be prevented by this work is a fork bomb conducted in a shell. We prevent it by noting that processes use a limited amount of stack pages. Seen this way, a fork bomb is just a special
case of resource abuse. If the offender is unable to grab more pages for the
stack, no new processes can be created.

There are also other things the general mechanism protects against. For
example, using too much of pinned dentry and inode cache, by touching files an
leaving them in memory forever.

In fact, a simple:

```
while true; do mkdir x; cd x; done
```
can halt your system easily because the file system limits are hard to reach
(big disks), but the kernel memory is not. Those are examples, but the list
certainly don't stop here.

An important use case for all that, is concerned with people offering hosting
services through containers. In a physical box we can put a limit to some
resources, like total number of processes or threads. But in an environment
where each independent user gets its own piece of the machine, we don't want a
potentially malicious user to destroy good users' services.

This might be true for systemd as well, that now groups services inside
cgroups. They generally want to put forward a set of guarantees that limits the
running service in a variety of ways, so that if they become badly behaved,
they won't interfere with the rest of the system.

There is, of course, a cost for that. To attempt to mitigate that, static
branches are used to make sure that even if the feature is compiled in with
potentially a lot of memory cgroups deployed this code will only be enabled
after the first user of this service configures any limit. Limits lower than
the user limit effectively means there is a separate kernel memory limit that
may be reached independently than the user limit. Values equal or greater than
the user limit implies only that kernel memory is tracked. This provides a
unified vision of "maximum memory", be it kernel or user memory. Because this
is all default-off, existing deployments will see no change in behavior.

Glauber Costa (12):
memcg: change defines to an enum
kmem accounting basic infrastructure
Add a __GFP_KMEMCG flag
memcg: kmem controller infrastructure
mm: Allocate kernel pages to the right memcg
res_counter: return amount of charges after res_counter_uncharge
memcg: kmem accounting lifecycle management
memcg: use static branches when code not in use
memcg: allow a memcg with kmem charges to be destructed.
execute the whole memcg freeing in free_worker
protect architectures where THREAD_SIZE >= PAGE_SIZE against fork
bombs
Add documentation about the kmem controller

Suleiman Souhlal (2):

memcg: Make it possible to use the stock for more than one page.
memcg: Reclaim when more than one page needed.

11 files changed, 717 insertions(+), 81 deletions(-)