Subject: [PATCH 00/10] memcg: Kernel Memory Accounting.
Posted by Suleiman Souhlal on Mon, 27 Feb 2012 22:58:43 GMT
View Forum Message <> Reply to Message

This patch series introduces kernel memory accounting to memcg.
It currently only accounts for slab.

It's very similar to the patchset I sent back in October, but
with a number of fixes and improvements.
There is also overlap with Glauber's patchset, but I decided to
send this because there are some pretty significant differences.

With this patchset, kernel memory gets counted in a memcg's
memory.kmem.usage_in_bytes.
It's possible to have a limit to the kernel memory by setting
memory.kmem.independent_kmem_limit to 1 and setting the limit in
memory.kmem.limit_in_bytes.
If memory.kmem.independent_kmem_limit is unset, kernel memory also
gets counted in the cgroup's memory.usage_in_bytes, and kernel
memory allocations may trigger memcg reclaim or OOM.

Slab gets accounted per-page, by using per-cgroup kmem_caches that
get created the first time an allocation of that type is done by
that cgroup.

The main difference with Glauber's patches is here: We try to
track all the slab allocations, while Glauber only tracks ones
that are explicitly marked.
We feel that it's important to track everything, because there
are a lot of different slab allocations that may use significant
amounts of memory, that we may not know of ahead of time.
This is also the main source of complexity in the patchset.

The per-cgroup kmem_cache approach makes it so that we only have
to do charges/uncharges in the slow path of the slab allocator,
which should have low performance impacts.

A per-cgroup kmem_cache will appear in slabinfo named like its
original cache, with the cgroup's name in parenthesis.
On cgroup deletion, the accounting gets moved to the root cgroup
and any existing cgroup kmem_cache gets "dead" appended to its
name, to indicate that its accounting was migrated.
The dead caches get removed once they no longer have any active
objects in them.

This patchset does not include any shrinker changes. We already have
patches for that, but I felt like it's more important to get the
accounting right first, before concentrating on the slab shrinking.

Some caveats:
- Only supports slab.c.
- There is a difference with non-memcg slab allocation in
  that with this, some slab allocations might fail when
  they never failed before. For example, a GFP_NOIO slab
  allocation wouldn't fail, but now it might.
  We have a change that makes slab accounting behave
  the same as non-accounted allocations, but I wasn't sure
  how important it was to include.
- We currently do two atomic operations on every accounted
  slab free, when we increment and decrement the kmem_cache's
  refcount. It's most likely possible to fix this.
- When CONFIG_CGROUP_MEM_RES_CTLR_KMEM is enabled, some
  conditional branches get added to the slab fast paths.
  That said, when the config option is disabled, this
  patchset is essentially a no-op.

I hope either this or Glauber's patchset will evolve into something
that is satisfactory to all the parties.

Patch series, based on Linus HEAD from today:

1/10 memcg: Kernel memory accounting infrastructure.
2/10 memcg: Uncharge all kmem when deleting a cgroup.
3/10 memcg: Reclaim when more than one page needed.
4/10 memcg: Introduce __GFP_NOACCOUNT.
5/10 memcg: Slab accounting.
6/10 memcg: Track all the memcg children of a kmem_cache.
7/10 memcg: Stop res_counter underflows.
8/10 memcg: Add CONFIG_CGROUP_MEM_RES_CTLR_KMEM_ACCT_ROOT.
9/10 memcg: Per-memcg memory.kmem.slabinfo file.
10/10 memcg: Document kernel memory accounting.

```
Documentation/cgroups/memory.txt |   44 +++-
include/linux/gfp.h              |    2 +
include/linux/memcontrol.h       |   30 ++-
include/linux/slab.h             |    1 +
include/linux/slab_def.h         |  102 +++++++-
init/Kconfig                     |    8 +
mm/memcontrol.c                  |  607 ++++++++++++++++++++++++++++++++++++--
mm/slab.c                        |  395 +++++++++++++++++++++---
8 files changed, 1115 insertions(+), 74 deletions(-)
```

-- Suleiman