Subject: [RFC] slub: show dead memcg caches in a separate file
Posted by Glauber Costa on Thu, 03 May 2012 18:47:21 GMT
View Forum Message <> Reply to Message

One of the very few things that still unsettles me in the kmem
controller for memcg, is how badly we mess up with the
/proc/slabinfo file.

It is alright to have the cgroup caches listed in slabinfo, but once
they die, I think they should be removed right away. A box full
of containers that come and go will rapidly turn that file into
a supreme mess. However, we currently leave them there so we can
determine where our used memory currently is.

This patch attempts to clean this up by creating a separate proc file
only to handle the dead slabs. Among other advantages, we need a lot
less information in a dead cache: only its current size in memory
matters to us.

So besides avoiding polution of the slabinfo files, we can access
dead cache information itself in a cleaner way.

I implemented this as a proof of concept while finishing up
my last round for submission. But I am sending this separately
to collect opinions from all of you. I can either implement
a version of this for the slab, or follow any other route.

Thanks

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
CC: Michal Hocko <mhocko@suse.cz>
CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
CC: Johannes Weiner <hannes@cmpxchg.org>
CC: Suleiman Souhlal <suleiman@google.com>
CC: Frederic Weisbecker <fweisbec@gmail.com>
---
 include/linux/slab.h | 3 ++
 mm/slub.c | 82 ++
 2 files changed, 85 insertions(+), 0 deletions(-)

diff --git a/include/linux/slab.h b/include/linux/slab.h
index 5df00c1..d6a0cf4 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -171,6 +171,9 @@ struct mem_cgroup_cache_params {
 #endif

```
  struct list_head destroyed_list; /* Used when deleting cpuset cache */
 };
+
+extern void memcg_finish_slab_destruction(void);
+extern void memcg_start_slab_destruction(void);
 #endif

 /*
diff --git a/mm/slub.c b/mm/slub.c
index 0652e99..de066e3 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -5657,6 +5657,11 @@ static int s_show(struct seq_file *m, void *p)

  s = list_entry(p, struct kmem_cache, list);

+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (s->memcg_params.dead)
+  return 0;
+#endif
+
  for_each_online_node(node) {
   struct kmem_cache_node *n = get_node(s, node);

@@ -5688,6 +5693,83 @@ static const struct seq_operations slabinfo_op = {
  .show = s_show,
 };

+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+static void *s_start_dead(struct seq_file *m, loff_t *pos)
+{
+ loff_t n = *pos;
+
+ down_read(&slub_lock);
+ if (!n) {
+  seq_puts(m, "slab_name                                  ");
+  seq_puts(m, "pagesperslab num_slabs");
+  seq_putc(m, '\n');
+ }
+
+ return seq_list_start(&slab_caches, *pos);
+}
+
+static int s_show_dead(struct seq_file *m, void *p)
+{
+ unsigned long nr_slabs = 0;
+ struct kmem_cache *s;
+ int node;
```

```
+
+ s = list_entry(p, struct kmem_cache, list);
+
+ if (!s->memcg_params.dead)
+  return 0;
+
+ for_each_online_node(node) {
+  struct kmem_cache_node *n = get_node(s, node);
+
+  if (!n)
+   continue;
+
+  nr_slabs += atomic_long_read(&n->nr_slabs);
+ }
+
+ seq_printf(m, "%-40s %11d", s->name, (1 << oo_order(s->oo)));
+ seq_printf(m, " %9lu", nr_slabs);
+ seq_putc(m, '\n');
+ return 0;
+}
+
+
+static const struct seq_operations dead_slabinfo_op = {
+ .start = s_start_dead,
+ .next = s_next,
+ .stop = s_stop,
+ .show = s_show_dead,
+};
+
+static int dead_slabinfo_open(struct inode *inode, struct file *file)
+{
+ return seq_open(file, &dead_slabinfo_op);
+}
+
+static const struct file_operations proc_dead_slabinfo_operations = {
+ .open  = dead_slabinfo_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = seq_release,
+};
+
+static atomic_t dead_memcg_counter;
+
+void memcg_start_slab_destruction(void)
+{
+ if (atomic_add_return(1, &dead_memcg_counter) == 1)
+  proc_create("dead_slabinfo", S_IRUSR, NULL,
+       &proc_dead_slabinfo_operations);
```

```
+}
+
+void memcg_finish_slab_destruction(void)
+{
+ if (atomic_dec_and_test(&dead_memcg_counter))
+  remove_proc_entry("dead_slabinfo", NULL);
+}
+#endif
+
 static int slabinfo_open(struct inode *inode, struct file *file)
 {
  return seq_open(file, &slabinfo_op);
--
1.7.7.6
```

Subject: Re: [RFC] slub: show dead memcg caches in a separate file
Posted by Suleiman Souhlal on Mon, 07 May 2012 22:04:39 GMT
View Forum Message <> Reply to Message

On Thu, May 3, 2012 at 11:47 AM, Glauber Costa <glommer@parallels.com> wrote:
> One of the very few things that still unsettles me in the kmem
> controller for memcg, is how badly we mess up with the
> /proc/slabinfo file.
>
> It is alright to have the cgroup caches listed in slabinfo, but once
> they die, I think they should be removed right away. A box full
> of containers that come and go will rapidly turn that file into
> a supreme mess. However, we currently leave them there so we can
> determine where our used memory currently is.
>
> This patch attempts to clean this up by creating a separate proc file
> only to handle the dead slabs. Among other advantages, we need a lot
> less information in a dead cache: only its current size in memory
> matters to us.
>
> So besides avoiding polution of the slabinfo files, we can access
> dead cache information itself in a cleaner way.
>
> I implemented this as a proof of concept while finishing up
> my last round for submission. But I am sending this separately
> to collect opinions from all of you. I can either implement
> a version of this for the slab, or follow any other route.

I don't really understand why the "dead" slabs are considered as
polluting slabinfo.

They still have objects in them, and I think that hiding them would

not be the right thing to do (even if they are available in a separate
file): They will incorrectly not be seen by programs like slabtop.

-- Suleiman

Subject: Re: [RFC] slub: show dead memcg caches in a separate file
Posted by Glauber Costa on Tue, 08 May 2012 03:30:16 GMT
View Forum Message <> Reply to Message

On 05/07/2012 07:04 PM, Suleiman Souhlal wrote:
> On Thu, May 3, 2012 at 11:47 AM, Glauber Costa<glommer@parallels.com> wrote:
>> One of the very few things that still unsettles me in the kmem
>> controller for memcg, is how badly we mess up with the
>> /proc/slabinfo file.
>>
>> It is alright to have the cgroup caches listed in slabinfo, but once
>> they die, I think they should be removed right away. A box full
>> of containers that come and go will rapidly turn that file into
>> a supreme mess. However, we currently leave them there so we can
>> determine where our used memory currently is.
>>
>> This patch attempts to clean this up by creating a separate proc file
>> only to handle the dead slabs. Among other advantages, we need a lot
>> less information in a dead cache: only its current size in memory
>> matters to us.
>>
>> So besides avoiding polution of the slabinfo files, we can access
>> dead cache information itself in a cleaner way.
>>
>> I implemented this as a proof of concept while finishing up
>> my last round for submission. But I am sending this separately
>> to collect opinions from all of you. I can either implement
>> a version of this for the slab, or follow any other route.
>
> I don't really understand why the "dead" slabs are considered as
> polluting slabinfo.
>
> They still have objects in them, and I think that hiding them would
> not be the right thing to do (even if they are available in a separate
> file): They will incorrectly not be seen by programs like slabtop.
>

Well, technically speaking, they aren't consider. I consider. The
difference is subtle, but boils down to if no one else consider this a
problem... there is no problem.

Now let me expand on the subject of why I do consider this unneeded

information (needed, just not here)

Consider a hosting box with ~100 caches. Let us say that a container
touches 50 of them, we still have 50 caches per container. Objects in
those caches, may take a long time to go away. Let's say, in 40 of those
caches.

The number of entries in /proc/slabinfo is not proportional to the
number of active containers: It becomes proportional to the number of
containers that *ever* existed on the machine - even if those numbers
drop with time, they still can drop slowly.

In use cases where containers come and go frequently, before a shrinker
can be called to wipe some of them out, we are easily in the 1000s of
lines in /proc/slabinfo. It becomes too much information, and it usually
makes it hard to find the one you are looking for.

But there is another aspect: those dead caches have one thing in common,
which is the fact that no new objects will ever be allocated on them.
You can't tune them, or do anything with them. I believe it is
misleading to include them in slabinfo.

The fact that the caches change names - to append "dead" may also break
tools, if that is what you are concerned about.

For all the above, I think a better semantics for slabinfo is to include
the active caches, and leave the dead ones somewhere else.

Subject: Re: [RFC] slub: show dead memcg caches in a separate file
Posted by Pekka Enberg on Tue, 08 May 2012 05:42:04 GMT
View Forum Message <> Reply to Message

On Tue, May 8, 2012 at 6:30 AM, Glauber Costa <glommer@parallels.com> wrote:
> But there is another aspect: those dead caches have one thing in common,
> which is the fact that no new objects will ever be allocated on them. You
> can't tune them, or do anything with them. I believe it is misleading to
> include them in slabinfo.
>
> The fact that the caches change names - to append "dead" may also break
> tools, if that is what you are concerned about.
>
> For all the above, I think a better semantics for slabinfo is to include the
> active caches, and leave the dead ones somewhere else.

Can these "dead caches" still hold on to physical memory? If so, they
must appear in /proc/slabinfo.

Subject: Re: [RFC] slub: show dead memcg caches in a separate file
Posted by Glauber Costa on Tue, 08 May 2012 11:55:14 GMT
View Forum Message <> Reply to Message

On 05/08/2012 02:42 AM, Pekka Enberg wrote:
> On Tue, May 8, 2012 at 6:30 AM, Glauber Costa<glommer@parallels.com> wrote:
>> But there is another aspect: those dead caches have one thing in common,
>> which is the fact that no new objects will ever be allocated on them. You
>> can't tune them, or do anything with them. I believe it is misleading to
>> include them in slabinfo.
>>
>> The fact that the caches change names - to append "dead" may also break
>> tools, if that is what you are concerned about.
>>
>> For all the above, I think a better semantics for slabinfo is to include the
>> active caches, and leave the dead ones somewhere else.
>
> Can these "dead caches" still hold on to physical memory? If so, they
> must appear in /proc/slabinfo.
Yes, if they didn't, I would show them nowhere, instead of in a separate
file.

But okay, that's why I sent a separate RFC for that part.
I will revert this behavior.