Subject: [PATCH 0/4] My contribution towards the end of populate()
Posted by Glauber Costa on Tue, 20 Mar 2012 16:50:52 GMT
View Forum Message <> Reply to Message

Tejun,

Let me know what you think.
I am providing the bugfix for the recently discussed bogus warning
in your tree, + the sock memcg bits. You should be able to get rid
of populate after that.

Let me know if there is any change you want done, and I'll adapt it.

Glauber Costa (4):
  don't trigger warning when d_subdirs is not empty.
  pass struct mem_cgroup instead of struct cgroup to socket memcg
  provide a function to register more cftype files into memcg
  get rid of populate for memcg

 include/linux/memcontrol.h | 1 +
 include/net/sock.h | 17 ++++++++++------
 include/net/tcp_memcontrol.h | 4 ++--
 kernel/cgroup.c | 8 +++++++-
 mm/memcontrol.c | 38 ++++++++++++++++++++++----------------
 net/core/sock.c | 15 +++++++++-----
 net/ipv4/tcp_memcontrol.c | 17 ++++++++++------
 7 files changed, 58 insertions(+), 42 deletions(-)

--
1.7.7.6

Subject: [PATCH 2/4] pass struct mem_cgroup instead of struct cgroup to socket memcg
Posted by Glauber Costa on Tue, 20 Mar 2012 16:50:54 GMT
View Forum Message <> Reply to Message

The only reason cgroup was used, was to be consistent with the populate()
interface. Now that we're getting rid of it, not only we no longer need
it, but we also *can't* call it this way.

Since we will no longer rely on populate(), this will be called from
create(). During create, the association between struct mem_cgroup
and struct cgroup does not yet exist, since cgroup internals hasn't
yet initialized its bookkeeping. This means we would not be able
to draw the memcg pointer from the cgroup pointer in these
functions, which is highly undesirable.

```
Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Tejun Heo <tj@kernel.org>
---
 include/net/sock.h        |   12 ++++++------
 mm/memcontrol.c           |   24 ++++++++++++++----------
 net/core/sock.c           |   10 +++++-----
 net/ipv4/tcp_memcontrol.c |    9 ++++-----
 4 files changed, 24 insertions(+), 31 deletions(-)

diff --git a/include/net/sock.h b/include/net/sock.h
index 705d1ad..e476277 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -67,16 +67,16 @@
 struct cgroup;
 struct cgroup_subsys;
 #ifdef CONFIG_NET
-int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys *ss);
-void mem_cgroup_sockets_destroy(struct cgroup *cgrp);
+int mem_cgroup_sockets_init(struct mem_cgroup *memcg, struct cgroup_subsys *ss);
+void mem_cgroup_sockets_destroy(struct mem_cgroup *memcg);
 #else
 static inline
-int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys *ss)
+int mem_cgroup_sockets_init(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
  return 0;
 }
 static inline
-void mem_cgroup_sockets_destroy(struct cgroup *cgrp)
+void mem_cgroup_sockets_destroy(struct mem_cgroup *memcg)
 {
 }
 #endif
@@ -867,9 +867,9 @@ struct proto {
   * This function has to setup any files the protocol want to
   * appear in the kmem cgroup filesystem.
   */
- int   (*init_cgroup)(struct cgroup *cgrp,
+ int   (*init_cgroup)(struct mem_cgroup *memcg,
                       struct cgroup_subsys *ss);
- void  (*destroy_cgroup)(struct cgroup *cgrp);
+ void  (*destroy_cgroup)(struct mem_cgroup *memcg);
  struct cg_proto  *(*proto_cgroup)(struct mem_cgroup *memcg);
 #endif
 };
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
```

```
index f2221ce..f7c6727 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -4568,29 +4568,22 @@ static int mem_control_numa_stat_open(struct inode *unused,
struct file *file)
 #endif /* CONFIG_NUMA */

 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
-static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
+static int register_kmem_files(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
- /*
-  * Part of this would be better living in a separate allocation
-  * function, leaving us with just the cgroup tree population work.
-  * We, however, depend on state such as network's proto_list that
-  * is only initialized after cgroup creation. I found the less
-  * cumbersome way to deal with it to defer it all to populate time
-  */
- return mem_cgroup_sockets_init(cont, ss);
+ return mem_cgroup_sockets_init(memcg, ss);
 };

-static void kmem_cgroup_destroy(struct cgroup *cont)
+static void kmem_cgroup_destroy(struct mem_cgroup *memcg)
 {
- mem_cgroup_sockets_destroy(cont);
+ mem_cgroup_sockets_destroy(memcg);
 }
 #else
-static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
+static int register_kmem_files(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
  return 0;
 }

-static void kmem_cgroup_destroy(struct cgroup *cont)
+static void kmem_cgroup_destroy(struct mem_cgroup *memcg)
 {
 }
 #endif
@@ -4947,7 +4940,7 @@ static void mem_cgroup_destroy(struct cgroup *cont)
 {
  struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);

- kmem_cgroup_destroy(cont);
+ kmem_cgroup_destroy(memcg);

  mem_cgroup_put(memcg);
```

```
 }
@@ -4955,7 +4948,8 @@ static void mem_cgroup_destroy(struct cgroup *cont)
 static int mem_cgroup_populate(struct cgroup_subsys *ss,
     struct cgroup *cont)
 {
- return register_kmem_files(cont, ss);
+ struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);
+ return register_kmem_files(memcg, ss);
 }

 #ifdef CONFIG_MMU
diff --git a/net/core/sock.c b/net/core/sock.c
index 688037c..443a404 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -141,7 +141,7 @@ static DEFINE_MUTEX(proto_list_mutex);
 static LIST_HEAD(proto_list);

 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
-int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys *ss)
+int mem_cgroup_sockets_init(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
  struct proto *proto;
  int ret = 0;
@@ -149,7 +149,7 @@ int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys
*ss)
  mutex_lock(&proto_list_mutex);
  list_for_each_entry(proto, &proto_list, node) {
   if (proto->init_cgroup) {
-   ret = proto->init_cgroup(cgrp, ss);
+   ret = proto->init_cgroup(memcg, ss);
    if (ret)
     goto out;
   }
@@ -160,19 +160,19 @@ int mem_cgroup_sockets_init(struct cgroup *cgrp, struct
cgroup_subsys *ss)
 out:
  list_for_each_entry_continue_reverse(proto, &proto_list, node)
   if (proto->destroy_cgroup)
-   proto->destroy_cgroup(cgrp);
+   proto->destroy_cgroup(memcg);
  mutex_unlock(&proto_list_mutex);
  return ret;
 }

-void mem_cgroup_sockets_destroy(struct cgroup *cgrp)
+void mem_cgroup_sockets_destroy(struct mem_cgroup *memcg)
 {
```

```
  struct proto *proto;

  mutex_lock(&proto_list_mutex);
  list_for_each_entry_reverse(proto, &proto_list, node)
   if (proto->destroy_cgroup)
-   proto->destroy_cgroup(cgrp);
+   proto->destroy_cgroup(memcg);
  mutex_unlock(&proto_list_mutex);
 }
 #endif
diff --git a/net/ipv4/tcp_memcontrol.c b/net/ipv4/tcp_memcontrol.c
index 21f48f3..94f32ce 100644
--- a/net/ipv4/tcp_memcontrol.c
+++ b/net/ipv4/tcp_memcontrol.c
@@ -51,7 +51,7 @@ static void memcg_tcp_enter_memory_pressure(struct sock *sk)
 }
 EXPORT_SYMBOL(memcg_tcp_enter_memory_pressure);

-int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
+int tcp_init_cgroup(struct mem_cgroup *memcg, struct cgroup_subsys *ss)
 {
  /*
   * The root cgroup does not use res_counters, but rather,
@@ -61,8 +61,7 @@ int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
  struct res_counter *res_parent = NULL;
  struct cg_proto *cg_proto, *parent_cg;
  struct tcp_memcontrol *tcp;
- struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
- struct mem_cgroup *parent = parent_mem_cgroup(memcg);
+ struct mem_cgroup *parent;
  struct net *net = current->nsproxy->net_ns;

  cg_proto = tcp_prot.proto_cgroup(memcg);
@@ -76,6 +75,7 @@ int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
  tcp->tcp_prot_mem[2] = net->ipv4.sysctl_tcp_mem[2];
  tcp->tcp_memory_pressure = 0;

+ parent = parent_mem_cgroup(memcg);
  parent_cg = tcp_prot.proto_cgroup(parent);
  if (parent_cg)
   res_parent = parent_cg->memory_allocated;
@@ -94,9 +94,8 @@ int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
 }
 EXPORT_SYMBOL(tcp_init_cgroup);

-void tcp_destroy_cgroup(struct cgroup *cgrp)
+void tcp_destroy_cgroup(struct mem_cgroup *memcg)
 {
```

```
- struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
  struct cg_proto *cg_proto;
  struct tcp_memcontrol *tcp;
  u64 val;
--
1.7.7.6
```

Subject: [PATCH 3/4] provide a function to register more cftype files into memcg
Posted by Glauber Costa on Tue, 20 Mar 2012 16:50:55 GMT
View Forum Message <> Reply to Message

The function mem_cgroup_register_cftype() is provided here, so
an optional memcg subsystem that needs to register files at
a time later than memcg initialization can do it.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Tejun Heo <tj@kernel.org>
CC: Aneesh Kumar K.V <aneesh.kumar@linux.vnet.ibm.com>
---
 include/linux/memcontrol.h | 1 +
 mm/memcontrol.c | 6 ++++++
 2 files changed, 7 insertions(+), 0 deletions(-)

```
diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index 4d34356..bcf8de3 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -141,6 +141,7 @@ static inline bool mem_cgroup_disabled(void)
  return false;
 }

+int mem_cgroup_register_cftype(const struct cftype *cfts);
 void mem_cgroup_update_page_stat(struct page *page,
      enum mem_cgroup_page_stat_item idx,
      int val);
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index f7c6727..bd91a46 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -444,6 +444,12 @@ struct cgroup_subsys_state *mem_cgroup_css(struct mem_cgroup
*memcg)
  return &memcg->css;
 }

+int mem_cgroup_register_cftype(const struct cftype *cfts)
+{
+ return WARN_ON(cgroup_add_cftypes(&mem_cgroup_subsys, cfts));
```

```
+}
+EXPORT_SYMBOL(mem_cgroup_register_cftype);
+
 static struct mem_cgroup_per_zone *
 page_cgroup_zoneinfo(struct mem_cgroup *memcg, struct page *page)
 {
--
1.7.7.6
```

Subject: [PATCH 4/4] get rid of populate for memcg
Posted by Glauber Costa on Tue, 20 Mar 2012 16:50:56 GMT
View Forum Message <> Reply to Message

The last man standing justifying the need for populate() is the
sock memcg initialization functions. By adopting the convention
that prot->init_cgroup is always called (when existing) with
NULL arguments, we are able to comunicate to the underlying protocol
that it needs to perform global initialization.

We can use that to register cftype files, and then follow the cgroup
rework that gets rid of populate().

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Tejun Heo <tj@kernel.org>

```
 include/net/sock.h          |    5 +++++
 include/net/tcp_memcontrol.h |    4 ++--
 mm/memcontrol.c             |   12 +++++---------
 net/core/sock.c             |    5 +++++
 net/ipv4/tcp_memcontrol.c   |    8 +++++++-
 5 files changed, 22 insertions(+), 12 deletions(-)

diff --git a/include/net/sock.h b/include/net/sock.h
index e476277..3daf7cd94 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -866,6 +866,11 @@ struct proto {
   * protocols that implement it, from cgroups populate function.
   * This function has to setup any files the protocol want to
   * appear in the kmem cgroup filesystem.
+  *
+  * init_cgroup() will be called upon protocol registering, with
+  * all arguments being NULL. The protocol should act accordingly,
+  * by doing global initialization in case any memcg is already
+  * initialized, like registering its files.
   */
  int   (*init_cgroup)(struct mem_cgroup *memcg,
```

```
            struct cgroup_subsys *ss);
diff --git a/include/net/tcp_memcontrol.h b/include/net/tcp_memcontrol.h
index 48410ff..7df18bc 100644
--- a/include/net/tcp_memcontrol.h
+++ b/include/net/tcp_memcontrol.h
@@ -12,8 +12,8 @@ struct tcp_memcontrol {
 };

 struct cg_proto *tcp_proto_cgroup(struct mem_cgroup *memcg);
-int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss);
-void tcp_destroy_cgroup(struct cgroup *cgrp);
+int tcp_init_cgroup(struct mem_cgroup *memcg, struct cgroup_subsys *ss);
+void tcp_destroy_cgroup(struct mem_cgroup *memcg);
 unsigned long long tcp_max_memory(const struct mem_cgroup *memcg);
 void tcp_prot_mem(struct mem_cgroup *memcg, long val, int idx);
 #endif /* _TCP_MEMCG_H */
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index bd91a46..87a1e21 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -4929,7 +4929,9 @@ mem_cgroup_create(struct cgroup *cont)
  atomic_set(&memcg->refcnt, 1);
  memcg->move_charge_at_immigrate = 0;
  mutex_init(&memcg->thresholds_lock);
- return &memcg->css;
+
+ if (!register_kmem_files(memcg, &mem_cgroup_subsys))
+  return &memcg->css;
 free_out:
  __mem_cgroup_free(memcg);
  return ERR_PTR(error);
@@ -4951,13 +4953,6 @@ static void mem_cgroup_destroy(struct cgroup *cont)
  mem_cgroup_put(memcg);
 }

-static int mem_cgroup_populate(struct cgroup_subsys *ss,
-    struct cgroup *cont)
-{
- struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);
- return register_kmem_files(memcg, ss);
-}
-
 #ifdef CONFIG_MMU
 /* Handlers for move charge at task migration. */
 #define PRECHARGE_COUNT_AT_ONCE 256
@@ -5465,7 +5460,6 @@ struct cgroup_subsys mem_cgroup_subsys = {
  .create = mem_cgroup_create,
  .pre_destroy = mem_cgroup_pre_destroy,
```

```
  .destroy = mem_cgroup_destroy,
- .populate = mem_cgroup_populate,
  .can_attach = mem_cgroup_can_attach,
  .cancel_attach = mem_cgroup_cancel_attach,
  .attach = mem_cgroup_move_task,
diff --git a/net/core/sock.c b/net/core/sock.c
index 443a404..1d60538 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -2484,6 +2484,11 @@ int proto_register(struct proto *prot, int alloc_slab)
   }
  }

+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (prot->init_cgroup)
+  prot->init_cgroup(NULL, NULL);
+#endif
+
  mutex_lock(&proto_list_mutex);
  list_add(&prot->node, &proto_list);
  assign_proto_idx(prot);
diff --git a/net/ipv4/tcp_memcontrol.c b/net/ipv4/tcp_memcontrol.c
index 94f32ce..b8cdf50 100644
--- a/net/ipv4/tcp_memcontrol.c
+++ b/net/ipv4/tcp_memcontrol.c
@@ -37,7 +37,6 @@ static struct cftype tcp_files[] = {
  },
  { } /* terminate */
 };
-CGROUP_SUBSYS_CFTYPES(mem_cgroup_subsys, tcp_files);

 static inline struct tcp_memcontrol *tcp_from_cgproto(struct cg_proto *cg_proto)
 {
@@ -64,6 +63,13 @@ int tcp_init_cgroup(struct mem_cgroup *memcg, struct cgroup_subsys
*ss)
  struct mem_cgroup *parent;
  struct net *net = current->nsproxy->net_ns;

+ if (!memcg && !ss)
+  return mem_cgroup_register_cftype(tcp_files);
+
+ if (WARN_ON(!memcg || !ss)) {
+  return -1;
+ }
+
  cg_proto = tcp_prot.proto_cgroup(memcg);
  if (!cg_proto)
   return 0;
```

--
1.7.7.6

Subject: Re: [PATCH 4/4] get rid of populate for memcg
Posted by Tejun Heo on Tue, 20 Mar 2012 18:31:49 GMT
View Forum Message <> Reply to Message

Hello, Glauber.

On Tue, Mar 20, 2012 at 08:50:56PM +0400, Glauber Costa wrote:
> @@ -4929,7 +4929,9 @@ mem_cgroup_create(struct cgroup *cont)
> atomic_set(&memcg->refcnt, 1);
> memcg->move_charge_at_immigrate = 0;
> mutex_init(&memcg->thresholds_lock);
> - return &memcg->css;
> +
> + if (!register_kmem_files(memcg, &mem_cgroup_subsys))
> + return &memcg->css;

After the change, I think register_kmem_files() is a quite misleading
name.

> @@ -2484,6 +2484,11 @@ int proto_register(struct proto *prot, int alloc_slab)
> }
> }
>
> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
> + if (prot->init_cgroup)
> + prot->init_cgroup(NULL, NULL);
> +#endif

So, init_cgroup() is overloaded to do two things - one load time init
and per-cgroup init, depending on the args.

> @@ -37,7 +37,6 @@ static struct cftype tcp_files[] = {
> },
> { } /* terminate */
> };
> -CGROUP_SUBSYS_CFTYPES(mem_cgroup_subsys, tcp_files);

What I don't get is why you can't just keep this. Is it because the
files might appear before the protocol is registered? Wouldn't it be
much better to add ipv4_tcp_init_cgroup() or whatever call to
inet_init() instead of overloading init_cgroup() with mostly unrelated
stuff?

Thanks.

--
tejun

Subject: Re: [PATCH 3/4] provide a function to register more cftype files into memcg
Posted by Tejun Heo on Tue, 20 Mar 2012 18:32:33 GMT
View Forum Message <> Reply to Message

Hey,

On Tue, Mar 20, 2012 at 08:50:55PM +0400, Glauber Costa wrote:
> The function mem_cgroup_register_cftype() is provided here, so
> an optional memcg subsystem that needs to register files at
> a time later than memcg initialization can do it.
>
> Signed-off-by: Glauber Costa <glommer@parallels.com>
> CC: Tejun Heo <tj@kernel.org>
> CC: Aneesh Kumar K.V <aneesh.kumar@linux.vnet.ibm.com>
> ---
> include/linux/memcontrol.h | 1 +
> mm/memcontrol.c | 6 ++++++
> 2 files changed, 7 insertions(+), 0 deletions(-)
>
> diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
> index 4d34356..bcf8de3 100644
> --- a/include/linux/memcontrol.h
> +++ b/include/linux/memcontrol.h
> @@ -141,6 +141,7 @@ static inline bool mem_cgroup_disabled(void)
>   return false;
> }
>
> +int mem_cgroup_register_cftype(const struct cftype *cfts);
> void mem_cgroup_update_page_stat(struct page *page,
>   enum mem_cgroup_page_stat_item idx,
>   int val);
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index f7c6727..bd91a46 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -444,6 +444,12 @@ struct cgroup_subsys_state *mem_cgroup_css(struct mem_cgroup *memcg)
>   return &memcg->css;
> }
>
> +int mem_cgroup_register_cftype(const struct cftype *cfts)
> +{

> + return WARN_ON(cgroup_add_cftypes(&mem_cgroup_subsys, cfts));
> +}
> +EXPORT_SYMBOL(mem_cgroup_register_cftype);

Why not just export mem_cgroup_subsys?

Thanks.

--
tejun

Subject: Re: [PATCH 3/4] provide a function to register more cftype files into memcg
Posted by Glauber Costa on Wed, 21 Mar 2012 07:27:12 GMT
View Forum Message <> Reply to Message

On 03/20/2012 10:32 PM, Tejun Heo wrote:
> Hey,
>
> On Tue, Mar 20, 2012 at 08:50:55PM +0400, Glauber Costa wrote:
>> The function mem_cgroup_register_cftype() is provided here, so
>> an optional memcg subsystem that needs to register files at
>> a time later than memcg initialization can do it.
>>
>> Signed-off-by: Glauber Costa<glommer@parallels.com>
>> CC: Tejun Heo<tj@kernel.org>
>> CC: Aneesh Kumar K.V<aneesh.kumar@linux.vnet.ibm.com>
>> ---
>> include/linux/memcontrol.h | 1 +
>> mm/memcontrol.c | 6 ++++++
>> 2 files changed, 7 insertions(+), 0 deletions(-)
>>
>> diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
>> index 4d34356..bcf8de3 100644
>> --- a/include/linux/memcontrol.h
>> +++ b/include/linux/memcontrol.h
>> @@ -141,6 +141,7 @@ static inline bool mem_cgroup_disabled(void)
>>   return false;
>> }
>>
>> +int mem_cgroup_register_cftype(const struct cftype *cfts);
>> void mem_cgroup_update_page_stat(struct page *page,
>>   enum mem_cgroup_page_stat_item idx,
>>   int val);
>> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
>> index f7c6727..bd91a46 100644
>> --- a/mm/memcontrol.c

>> +++ b/mm/memcontrol.c
>> @@ -444,6 +444,12 @@ struct cgroup_subsys_state *mem_cgroup_css(struct mem_cgroup
*memcg)
>> return&memcg->css;
>> }
>>
>> +int mem_cgroup_register_cftype(const struct cftype *cfts)
>> +{
>> + return WARN_ON(cgroup_add_cftypes(&mem_cgroup_subsys, cfts));
>> +}
>> +EXPORT_SYMBOL(mem_cgroup_register_cftype);
>
> Why not just export mem_cgroup_subsys?
>

I'm fine either way. I usually prefer not exporting raw data like this,
but that's 100 % taste. How do you prefer me to do it?

Subject: Re: [PATCH 4/4] get rid of populate for memcg
Posted by Glauber Costa on Wed, 21 Mar 2012 07:36:19 GMT
View Forum Message <> Reply to Message

On 03/20/2012 10:31 PM, Tejun Heo wrote:
> Hello, Glauber.
>
> On Tue, Mar 20, 2012 at 08:50:56PM +0400, Glauber Costa wrote:
>> @@ -4929,7 +4929,9 @@ mem_cgroup_create(struct cgroup *cont)
>> atomic_set(&memcg->refcnt, 1);
>> memcg->move_charge_at_immigrate = 0;
>> mutex_init(&memcg->thresholds_lock);
>> - return&memcg->css;
>> +
>> + if (!register_kmem_files(memcg,&mem_cgroup_subsys))
>> + return&memcg->css;
>
> After the change, I think register_kmem_files() is a quite misleading
> name.

how about init_kmem() ?

Remember the slab bits will are likely to end up here as well in the end.

>> @@ -2484,6 +2484,11 @@ int proto_register(struct proto *prot, int alloc_slab)
>> }
>> }
>>
>> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM

>> + if (prot->init_cgroup)
>> + prot->init_cgroup(NULL, NULL);
>> +#endif
>
> So, init_cgroup() is overloaded to do two things - one load time init
> and per-cgroup init, depending on the args.

Yes. I don't love it, but there is quite a bunch of precedents for this.
Like the shrinkers in vmscan, for instance.

a NULL argument is a probe, a valid argument should have action taken.

>> @@ -37,7 +37,6 @@ static struct cftype tcp_files[] = {
>>   },
>>   { } /* terminate */
>>   };
>> -CGROUP_SUBSYS_CFTYPES(mem_cgroup_subsys, tcp_files);
>
> What I don't get is why you can't just keep this. Is it because the
> files might appear before the protocol is registered? Wouldn't it be
> much better to add ipv4_tcp_init_cgroup() or whatever call to
> inet_init() instead of overloading init_cgroup() with mostly unrelated
> stuff?
>

The reason is that this has to be kept generic for protocols that may
want to implement this in the future - since the pressure controls
themselves are generic, the per-cgroup versions should be as well.

And in general, a protocol can live in a module, or not be registered
despite being compiled in.

When the root memcg is created, prot_register() is usually not yet
called, at least for tcp.

Now, what we do with the files, are our decision in the end. If you
want, we can use CGROUP_SUBSYS_CFTYPES(mem_cgroup_subsys, tcp_files)
as you suggested. tcp itself is always available if it is compiled in.
Then in the future, if anyone cares about adding support for a protocol
that may differ in that aspect, we can put the files nevertheless, and
use ENOTSUPP as kame suggested for the swap accounting.

What's your take ?

Subject: Re: [PATCH 4/4] get rid of populate for memcg
Posted by Tejun Heo on Wed, 21 Mar 2012 16:06:10 GMT

Hello, Glauber.

On Wed, Mar 21, 2012 at 11:36:19AM +0400, Glauber Costa wrote:
> On 03/20/2012 10:31 PM, Tejun Heo wrote:
> >Hello, Glauber.
> >
> >On Tue, Mar 20, 2012 at 08:50:56PM +0400, Glauber Costa wrote:
> >>@@ -4929,7 +4929,9 @@ mem_cgroup_create(struct cgroup *cont)
> >> atomic_set(&memcg->refcnt, 1);
> >> memcg->move_charge_at_immigrate = 0;
> >> mutex_init(&memcg->thresholds_lock);
> >>- return&memcg->css;
> >>+
> >>+ if (!register_kmem_files(memcg,&mem_cgroup_subsys))
> >>+ return&memcg->css;
> >
> >After the change, I think register_kmem_files() is a quite misleading
> >name.
>
> how about init_kmem() ?
>
> Remember the slab bits will are likely to end up here as well in the end.

I don't know. Whatever which describes what's going on.
memcg_init_kmem()?

> >So, init_cgroup() is overloaded to do two things - one load time init
> >and per-cgroup init, depending on the args.
>
> Yes. I don't love it, but there is quite a bunch of precedents for this.
> Like the shrinkers in vmscan, for instance.
>
> a NULL argument is a probe, a valid argument should have action taken.

Please don't. Just add a new callback if necessary.

> >What I don't get is why you can't just keep this. Is it because the
> >files might appear before the protocol is registered? Wouldn't it be
> >much better to add ipv4_tcp_init_cgroup() or whatever call to
> >inet_init() instead of overloading init_cgroup() with mostly unrelated
> >stuff?
> >
>
> The reason is that this has to be kept generic for protocols that
> may want to implement this in the future - since the pressure
> controls themselves are generic, the per-cgroup versions should be
> as well.

>
> And in general, a protocol can live in a module, or not be registered
> despite being compiled in.

Hmmmm... yeah, CGROUP_SUBSYS_CFTYPES() would register the files on
module load but won't unregister them on unload. Will fix that.
However, the fact that files living in modules shouldn't be a problem
in itself. If those file handlers can cope with protocol not being
registered yet, everything should be fine.

> Now, what we do with the files, are our decision in the end. If you
> want, we can use CGROUP_SUBSYS_CFTYPES(mem_cgroup_subsys, tcp_files)
> as you suggested. tcp itself is always available if it is compiled in.
> Then in the future, if anyone cares about adding support for a
> protocol that may differ in that aspect, we can put the files
> nevertheless, and
> use ENOTSUPP as kame suggested for the swap accounting.

I don't quite get why a protocol module would be loaded but not
reigstered. Do we actually have cases like that? I know it's
mechanically possible but don't think there's any actual use case or
existing code which does that, so no need to worry about them.

Thanks.

--
tejun

Subject: Re: [PATCH 3/4] provide a function to register more cftype files into memcg
Posted by Tejun Heo on Wed, 21 Mar 2012 16:11:22 GMT
View Forum Message <> Reply to Message

On Wed, Mar 21, 2012 at 11:27:12AM +0400, Glauber Costa wrote:
> >>+int mem_cgroup_register_cftype(const struct cftype *cfts)
> >>+{
> >>+ return WARN_ON(cgroup_add_cftypes(&mem_cgroup_subsys, cfts));
> >>+}
> >>+EXPORT_SYMBOL(mem_cgroup_register_cftype);
> >
> >Why not just export mem_cgroup_subsys?
> >
>
> I'm fine either way. I usually prefer not exporting raw data like
> this, but that's 100 % taste. How do you prefer me to do it?

I think exporting subsys directly is better than implementing thin

wrapper like above. IMHO, wrappers like above don't add any
functionality and are likely to just obfuscate what's going on.

Thanks.

--
tejun

Subject: Re: [PATCH 4/4] get rid of populate for memcg
Posted by Tejun Heo on Wed, 21 Mar 2012 16:19:03 GMT
View Forum Message <> Reply to Message

On Wed, Mar 21, 2012 at 09:06:10AM -0700, Tejun Heo wrote:
> I don't quite get why a protocol module would be loaded but not
> reigstered. Do we actually have cases like that? I know it's
> mechanically possible but don't think there's any actual use case or
> existing code which does that, so no need to worry about them.

Also, if a proto is registered from a module, it's gotta have a
module_init() which registers proto, right? Then, the right thing to
do would be just "register proto; register cftypes;" in the function
and the reverse of that in module_exit().

--
tejun

Subject: Re: [PATCH 4/4] get rid of populate for memcg
Posted by Glauber Costa on Wed, 21 Mar 2012 16:30:07 GMT
View Forum Message <> Reply to Message

On 03/21/2012 08:06 PM, Tejun Heo wrote:
> I don't quite get why a protocol module would be loaded but not
> reigstered. Do we actually have cases like that? I know it's
> mechanically possible but don't think there's any actual use case or
> existing code which does that, so no need to worry about them.
>

Probably I'm just over worrying here. If your macro can handle
unregistering well, I think I should be fine using them.

Subject: Re: [PATCH 3/4] provide a function to register more cftype files into memcg
Posted by Glauber Costa on Wed, 21 Mar 2012 16:30:28 GMT

On 03/21/2012 08:11 PM, Tejun Heo wrote:
>> I'm fine either way. I usually prefer not exporting raw data like
>> > this, but that's 100 % taste. How do you prefer me to do it?
> I think exporting subsys directly is better than implementing thin
> wrapper like above. IMHO, wrappers like above don't add any
> functionality and are likely to just obfuscate what's going on.
>
> Thanks.

Ok, I can change that.