Subject: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by Balbir Singh on Thu, 19 Jun 2008 03:13:43 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> I used memrlimit cgroup at the first time.
>
> May I ask a question about memrlimit cgroup ?
>
> In following
> ==
> static void memrlimit_cgroup_move_task(struct cgroup_subsys *ss,
>   struct cgroup *cgrp,
>   struct cgroup *old_cgrp,
>   struct task_struct *p)
> {
>   struct mm_struct *mm;
>   struct memrlimit_cgroup *memrcg, *old_memrcg;
>
> <snip>
>   if (res_counter_charge(&memrcg->as_res, (mm->total_vm << PAGE_SHIFT)))
>   goto out;
>   res_counter_uncharge(&old_memrcg->as_res, (mm->total_vm << PAGE_SHIFT));
> ==
> This is a callback for task_attach(). and this never fails.
>
> What happens when the moved task, which move-of-charge fails, exits ?
>

Good question - I am working on this, some of the logic should move to
can_attach(). I'll try and experiment with it and send out a fix.

> ==
> % mkdir /dev/cgroup/memrlimit/group_01
> % mkdir /dev/cgroup/memrlimit/group_02
> % echo 1G > /dev/cgroup/memrlimit/group_01/memrlimit.limit_in_bytes
> % echo 0 > /dev/cgroup/memrlimit/group_02/memrlimit.limit_in_bytes
> % echo $$ > /dev/cgroup/memrlimit/group_01/tasks
> % echo $$ > /dev/cgroup/memrlimit/group_02/tasks
> % exit
> == you'll see WARNING ==
>
> I think the charge of the new group goes to minus. right ?
> (and old group's charge never goes down.)
> I don't think this is "no problem".
>
> What kind of patch is necessary to fix this ?
> task_attach() should be able to fail in future ?

>
> I'm sorry if I misunderstand something or this is already in TODO list.
>

It's already on the TODO list. Thanks for keeping me reminded about it.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by KAMEZAWA Hiroyuki on Thu, 19 Jun 2008 03:14:35 GMT
View Forum Message <> Reply to Message

I used memrlimit cgroup at the first time.

May I ask a question about memrlimit cgroup ?

In following
==
```
static void memrlimit_cgroup_move_task(struct cgroup_subsys *ss,
                                struct cgroup *cgrp,
                                struct cgroup *old_cgrp,
                                struct task_struct *p)
{
        struct mm_struct *mm;
        struct memrlimit_cgroup *memrcg, *old_memrcg;

<snip>
        if (res_counter_charge(&memrcg->as_res, (mm->total_vm << PAGE_SHIFT)))
                goto out;
        res_counter_uncharge(&old_memrcg->as_res, (mm->total_vm << PAGE_SHIFT));
```
==
This is a callback for task_attach(). and this never fails.

What happens when the moved task, which move-of-charge fails, exits ?

==
```
% mkdir /dev/cgroup/memrlimit/group_01
% mkdir /dev/cgroup/memrlimit/group_02
% echo 1G > /dev/cgroup/memrlimit/group_01/memrlimit.limit_in_bytes
```

% echo 0 > /dev/cgroup/memrlimit/group_02/memrlimit.limit_in_bytes
% echo $$ > /dev/cgroup/memrlimit/group_01/tasks
% echo $$ > /dev/cgroup/memrlimit/group_02/tasks
% exit
== you'll see WARNING ==

I think the charge of the new group goes to minus. right ?
(and old group's charge never goes down.)
I don't think this is "no problem".

What kind of patch is necessary to fix this ?
task_attach() should be able to fail in future ?

I'm sorry if I misunderstand something or this is already in TODO list.

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by KAMEZAWA Hiroyuki on Thu, 19 Jun 2008 03:21:43 GMT
View Forum Message <> Reply to Message

On Thu, 19 Jun 2008 08:43:43 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

>
> > I think the charge of the new group goes to minus. right ?
> > (and old group's charge never goes down.)
> > I don't think this is "no problem".
> >
> > What kind of patch is necessary to fix this ?
> > task_attach() should be able to fail in future ?
> >
> > I'm sorry if I misunderstand something or this is already in TODO list.
> >
>
> It's already on the TODO list. Thanks for keeping me reminded about it.
>
Okay, I'm looking foward to see how can_attach and roll-back(if necessary)
is implemnted.
As you know, I'm interested in how to handle failure of task move.

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by KAMEZAWA Hiroyuki on Thu, 19 Jun 2008 10:19:05 GMT
View Forum Message <> Reply to Message

On Thu, 19 Jun 2008 12:24:29 +0900
KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> On Thu, 19 Jun 2008 08:43:43 +0530
> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> >
> > > I think the charge of the new group goes to minus. right ?
> > > (and old group's charge never goes down.)
> > > I don't think this is "no problem".
> > >
> > > What kind of patch is necessary to fix this ?
> > > task_attach() should be able to fail in future ?
> > >
> > > I'm sorry if I misunderstand something or this is already in TODO list.
> > >
> >
> > It's already on the TODO list. Thanks for keeping me reminded about it.
> >
> Okay, I'm looking foward to see how can_attach and roll-back(if necessary)
> is implemnted.
> As you know, I'm interested in how to handle failure of task move.
>
One more thing...
Now, charge is done at

 - vm is inserted (special case?)
 - vm is expanded (mmap is called, stack growth...)

And uncharge is done at
 - vm is removed (success of munmap)
 - exit_mm is called (exit of process)

But it seems charging at may_expand_vm() is not good.

The mmap can fail after may_expand_vm() because of various reason,
but charge is already done at may_expand_vm()....and no roll-back.

== an easy example of leak in stack growth handling ==
[root@iridium kamezawa]# cat /opt/cgroup/test/memrlimit.usage_in_bytes
71921664
[root@iridium kamezawa]# ulimit -s 3
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ulimit -s unlimited
[root@iridium kamezawa]# cat /opt/cgroup/test/memrlimit.usage_in_bytes
72368128
[root@iridium kamezawa]#
==

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by Balbir Singh on Thu, 19 Jun 2008 12:30:24 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> On Thu, 19 Jun 2008 12:24:29 +0900
> KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:
>
>> On Thu, 19 Jun 2008 08:43:43 +0530
>> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>>
>>>> I think the charge of the new group goes to minus. right ?
>>>> (and old group's charge never goes down.)
>>>> I don't think this is "no problem".
>>>>

>>>> What kind of patch is necessary to fix this ?
>>>> task_attach() should be able to fail in future ?
>>>>
>>>> I'm sorry if I misunderstand something or this is already in TODO list.
>>>>
>>> It's already on the TODO list. Thanks for keeping me reminded about it.
>>>
>> Okay, I'm looking foward to see how can_attach and roll-back(if necessary)
>> is implemnted.
>> As you know, I'm interested in how to handle failure of task move.
>>
> One more thing...
> Now, charge is done at
>
> - vm is inserted (special case?)
> - vm is expanded (mmap is called, stack growth...)
>
> And uncharge is done at
> - vm is removed (success of munmap)
> - exit_mm is called (exit of process)
>
> But it seems charging at may_expand_vm() is not good.
> The mmap can fail after may_expand_vm() because of various reason,
> but charge is already done at may_expand_vm()....and no roll-back.
>
> == an easy example of leak in stack growth handling ==
> [root@iridium kamezawa]# cat /opt/cgroup/test/memrlimit.usage_in_bytes
> 71921664
> [root@iridium kamezawa]# ulimit -s 3
> [root@iridium kamezawa]# ls
> Killed
> [root@iridium kamezawa]# ls
> Killed
> [root@iridium kamezawa]# ls
> Killed
> [root@iridium kamezawa]# ls
> Killed
> [root@iridium kamezawa]# ls
> Killed
> [root@iridium kamezawa]# ulimit -s unlimited
> [root@iridium kamezawa]# cat /opt/cgroup/test/memrlimit.usage_in_bytes
> 72368128
> [root@iridium kamezawa]#

Aaah.. I see.. I had it in place earlier, but moved them to may_expand_vm() on review suggestions. I can move it out or try to unroll when things fail. I'll experiment a bit more. Is there any particular method you prefer?

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by KAMEZAWA Hiroyuki on Thu, 19 Jun 2008 13:38:29 GMT
View Forum Message <> Reply to Message

----- Original Message -----
>Date: Thu, 19 Jun 2008 18:00:24 +0530
>From: Balbir Singh <balbir@linux.vnet.ibm.com>

>> [root@iridium kamezawa]# ulimit -s unlimited
>> [root@iridium kamezawa]# cat /opt/cgroup/test/memrlimit.usage_in_bytes
>> 72368128
>> [root@iridium kamezawa]#
>
>Aaah.. I see.. I had it in place earlier, but moved them to may_expand_vm() o
n
>review suggestions. I can move it out or try to unroll when things fail. I'll
>experiment a bit more. Is there any particular method you prefer?
>
Anywhere... but...IMHO, where the rlimit does charge will be a candidate.
But doing that may make the code ugly, I'm not sure now.

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by Balbir Singh on Thu, 19 Jun 2008 16:41:28 GMT
View Forum Message <> Reply to Message

* KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> [2008-06-19 19:22:27]:

> On Thu, 19 Jun 2008 12:24:29 +0900
> KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:
>
> > On Thu, 19 Jun 2008 08:43:43 +0530
> > Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> >
> > >
> > > > I think the charge of the new group goes to minus. right ?
> > > > (and old group's charge never goes down.)
> > > > I don't think this is "no problem".
> > > >
> > > > What kind of patch is necessary to fix this ?
> > > > task_attach() should be able to fail in future ?
> > > >
> > > > I'm sorry if I misunderstand something or this is already in TODO list.
> > > >
> > >
> > > It's already on the TODO list. Thanks for keeping me reminded about it.
> > >
> > Okay, I'm looking foward to see how can_attach and roll-back(if necessary)
> > is implemnted.
> > As you know, I'm interested in how to handle failure of task move.
> >
> One more thing...
> Now, charge is done at
>
> - vm is inserted (special case?)
> - vm is expanded (mmap is called, stack growth...)
>
> And uncharge is done at
> - vm is removed (success of munmap)
> - exit_mm is called (exit of process)
>
> But it seems charging at may_expand_vm() is not good.
> The mmap can fail after may_expand_vm() because of various reason,
> but charge is already done at may_expand_vm()....and no roll-back.
>
Hi, Kamezawa-San,

The patch I have below addresses the issue. FYI: I do see some
variation in memrlimit.usage_in_bytes after running ls, but it's not
that much. I verified that the patch behaves correctly, by doing
a cat of memrlimit.usage_in_bytes from another shell (not associated
with test group) and then compared that value to /proc/$$/statm
($$ being the pid of the task belonging to the test group).

Could you please review/test the patch below? If it works OK, I'll
request Andrew to pick it up.

Description
---
memrlimit cgroup does not handle error cases after may_expand_vm(). This
BUG was reported by Kamezawa, with the test case below to reproduce it

[root@iridium kamezawa]# cat /opt/cgroup/test/memrlimit.usage_in_bytes
71921664
[root@iridium kamezawa]# ulimit -s 3
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ls
Killed
[root@iridium kamezawa]# ulimit -s unlimited
[root@iridium kamezawa]# cat /opt/cgroup/test/memrlimit.usage_in_bytes
72368128
[root@iridium kamezawa]#

This patch adds better handling support to fix the reported problem.

Reported-By: kamezawa.hiroyu@jp.fujitsu.com
Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
---

 mm/mmap.c | 36 ++++++++++++++++++++++++++----------
 mm/mremap.c | 6 ++++++
 2 files changed, 31 insertions(+), 11 deletions(-)

diff -puN mm/mmap.c~memrlimit-cgroup-add-better-error-handling mm/mmap.c
--- linux-2.6.26-rc5/mm/mmap.c~memrlimit-cgroup-add-better-error-handling 2008-06-19
21:12:46.000000000 +0530
+++ linux-2.6.26-rc5-balbir/mm/mmap.c 2008-06-19 21:39:45.000000000 +0530
@@ -1123,7 +1123,7 @@ munmap_back:
     */
    charged = len >> PAGE_SHIFT;
    if (security_vm_enough_memory(charged))
-    return -ENOMEM;
+   goto undo_charge;
    vm_flags |= VM_ACCOUNT;
   }
  }
@@ -1245,6 +1245,8 @@ free_vma:

```
 unacct_error:
  if (charged)
   vm_unacct_memory(charged);
+undo_charge:
+ memrlimit_cgroup_uncharge_as(mm, len >> PAGE_SHIFT);
  return error;
 }

@@ -1540,14 +1542,15 @@ static int acct_stack_growth(struct vm_a
  struct mm_struct *mm = vma->vm_mm;
  struct rlimit *rlim = current->signal->rlim;
  unsigned long new_start;
+ int ret = -ENOMEM;

  /* address space limit tests */
  if (!may_expand_vm(mm, grow))
-  return -ENOMEM;
+  goto out;

  /* Stack limit test */
  if (size > rlim[RLIMIT_STACK].rlim_cur)
-  return -ENOMEM;
+  goto undo_charge;

  /* mlock limit tests */
  if (vma->vm_flags & VM_LOCKED) {
@@ -1556,21 +1559,23 @@ static int acct_stack_growth(struct vm_a
   locked = mm->locked_vm + grow;
   limit = rlim[RLIMIT_MEMLOCK].rlim_cur >> PAGE_SHIFT;
   if (locked > limit && !capable(CAP_IPC_LOCK))
-   return -ENOMEM;
+   goto undo_charge;
  }

  /* Check to ensure the stack will not grow into a hugetlb-only region */
  new_start = (vma->vm_flags & VM_GROWSUP) ? vma->vm_start :
   vma->vm_end - size;
- if (is_hugepage_only_range(vma->vm_mm, new_start, size))
-  return -EFAULT;
+ if (is_hugepage_only_range(vma->vm_mm, new_start, size)) {
+  ret = -EFAULT;
+  goto undo_charge;
+ }

  /*
   * Overcommit..  This must be the final test, as it will
   * update security statistics.
   */
```

```
  if (security_vm_enough_memory(grow))
-  return -ENOMEM;
+  goto undo_charge;

  /* Ok, everything looks good - let it rip */
  mm->total_vm += grow;
@@ -1578,6 +1583,11 @@ static int acct_stack_growth(struct vm_a
   mm->locked_vm += grow;
  vm_stat_account(mm, vma->vm_flags, vma->vm_file, grow);
  return 0;
+undo_charge:
+ /* Undo memrlimit charge */
+ memrlimit_cgroup_uncharge_as(mm, grow);
+out:
+ return ret;
 }

 #if defined(CONFIG_STACK_GROWSUP) || defined(CONFIG_IA64)
@@ -1982,6 +1992,7 @@ unsigned long do_brk(unsigned long addr,
  struct rb_node ** rb_link, * rb_parent;
  pgoff_t pgoff = addr >> PAGE_SHIFT;
  int error;
+ int ret = -ENOMEM;

  len = PAGE_ALIGN(len);
  if (!len)
@@ -2035,13 +2046,13 @@ unsigned long do_brk(unsigned long addr,

  /* Check against address space limits *after* clearing old maps... */
  if (!may_expand_vm(mm, len >> PAGE_SHIFT))
-  return -ENOMEM;
+  return ret;

  if (mm->map_count > sysctl_max_map_count)
-  return -ENOMEM;
+  goto undo_charge;

  if (security_vm_enough_memory(len >> PAGE_SHIFT))
-  return -ENOMEM;
+  goto undo_charge;

  /* Can we just expand an old private anonymous mapping? */
  vma = vma_merge(mm, prev, addr, addr + len, flags,
@@ -2055,7 +2066,7 @@ unsigned long do_brk(unsigned long addr,
  vma = kmem_cache_zalloc(vm_area_cachep, GFP_KERNEL);
  if (!vma) {
   vm_unacct_memory(len >> PAGE_SHIFT);
-  return -ENOMEM;
```

```
+  goto undo_charge;
  }

  vma->vm_mm = mm;
@@ -2073,6 +2084,9 @@ out:
    mm->locked_vm += (len >> PAGE_SHIFT) - nr_pages;
  }
  return addr;
+undo_charge:
+ memrlimit_cgroup_uncharge_as(mm, len >> PAGE_SHIFT);
+ return ret;
 }

 EXPORT_SYMBOL(do_brk);
diff -puN mm/mremap.c~memrlimit-cgroup-add-better-error-handling mm/mremap.c
--- linux-2.6.26-rc5/mm/mremap.c~memrlimit-cgroup-add-better-error-handling 2008-06-19
21:12:46.000000000 +0530
+++ linux-2.6.26-rc5-balbir/mm/mremap.c 2008-06-19 22:00:02.000000000 +0530
@@ -18,6 +18,7 @@
 #include <linux/highmem.h>
 #include <linux/security.h>
 #include <linux/syscalls.h>
+#include <linux/memrlimitcgroup.h>

 #include <asm/uaccess.h>
 #include <asm/cacheflush.h>
@@ -256,6 +257,7 @@ unsigned long do_mremap(unsigned long ad
  struct vm_area_struct *vma;
  unsigned long ret = -EINVAL;
  unsigned long charged = 0;
+ int vm_expanded = 0;

  if (flags & ~(MREMAP_FIXED | MREMAP_MAYMOVE))
   goto out;
@@ -349,6 +351,7 @@ unsigned long do_mremap(unsigned long ad
   goto out;
  }

+ vm_expanded = 1;
  if (vma->vm_flags & VM_ACCOUNT) {
   charged = (new_len - old_len) >> PAGE_SHIFT;
   if (security_vm_enough_memory(charged))
@@ -411,6 +414,9 @@ out:
  if (ret & ~PAGE_MASK)
   vm_unacct_memory(charged);
 out_nc:
+ if (vm_expanded)
+  memrlimit_cgroup_uncharge_as(mm,
```

```
+    (new_len - old_len) >> PAGE_SHIFT);
  return ret;
 }


_
--
 Warm Regards,
 Balbir Singh
 Linux Technology Center
 IBM, ISTL
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by Balbir Singh on Thu, 19 Jun 2008 18:25:56 GMT
View Forum Message <> Reply to Message

* KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> [2008-06-19 12:14:35]:

> I used memrlimit cgroup at the first time.
>
> May I ask a question about memrlimit cgroup ?
>

Hi, Kamezawa-San,

Could you please review/test the patch below to see if it solves your
problem? If it does, I'll push it up to Andrew

Description

This patch fixes a task migration problem reported by Kamezawa-San. This
patch should fix all issues with migraiton, except for a rare condition
documented in memrlimit_cgroup_move_task(). To fix that problem, we
would need to add transaction properties to cgroups.

The problem reported was that migrating to a group that did not have
sufficient limits to accept an incoming task caused a kernel warning.

Steps to reproduce

```
% mkdir /dev/cgroup/memrlimit/group_01
% mkdir /dev/cgroup/memrlimit/group_02
% echo 1G > /dev/cgroup/memrlimit/group_01/memrlimit.limit_in_bytes
```

```
% echo 0 >  /dev/cgroup/memrlimit/group_02/memrlimit.limit_in_bytes
% echo $$ > /dev/cgroup/memrlimit/group_01/tasks
% echo $$ > /dev/cgroup/memrlimit/group_02/tasks
% exit
```

memrlimit does the right thing by not moving the charges to group_02,
but the task is still put into g2 (since we did not use can_attach to
fail migration). Once in g2, when we echo the task to the root cgroup,
it tries to uncharge the cost of the task from g2. g2 does not have
any charge associated with the task, hence we get a warning.

Reported-by: kamezawa.hiroyu@jp.fujitsu.com
Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>

```
 include/linux/res_counter.h |   18 ++++++++++++++++++
 mm/memrlimitcgroup.c        |   40 ++++++++++++++++++++++++++++++++++++++++
 2 files changed, 58 insertions(+)

diff -puN mm/memrlimitcgroup.c~memrlimit-cgroup-fix-attach-task mm/memrlimitcgroup.c
--- linux-2.6.26-rc5/mm/memrlimitcgroup.c~memrlimit-cgroup-fix-attach-task 2008-06-19
22:17:46.000000000 +0530
+++ linux-2.6.26-rc5-balbir/mm/memrlimitcgroup.c 2008-06-19 23:48:27.000000000 +0530
@@ -166,6 +166,39 @@ static int memrlimit_cgroup_populate(str
     ARRAY_SIZE(memrlimit_cgroup_files));
 }

+static int memrlimit_cgroup_can_move_task(struct cgroup_subsys *ss,
+     struct cgroup *cgrp,
+     struct task_struct *p)
+{
+ struct mm_struct *mm;
+ struct memrlimit_cgroup *memrcg;
+ int ret = 0;
+
+ mm = get_task_mm(p);
+ if (mm == NULL)
+  return -EINVAL;
+
+ /*
+  * Hold mmap_sem, so that total_vm does not change underneath us
+  */
+ down_read(&mm->mmap_sem);
+
+ rcu_read_lock();
+ if (p != rcu_dereference(mm->owner))
+  goto out;
+
```

```
+ memrcg = memrlimit_cgroup_from_cgrp(cgrp);
+
+ if (!res_counter_add_check(&memrcg->as_res,
+     (mm->total_vm << PAGE_SHIFT)))
+  ret = -ENOMEM;
+out:
+ rcu_read_unlock();
+ up_read(&mm->mmap_sem);
+ mmput(mm);
+ return ret;
+}
+
 static void memrlimit_cgroup_move_task(struct cgroup_subsys *ss,
       struct cgroup *cgrp,
       struct cgroup *old_cgrp,
@@ -193,6 +226,12 @@ static void memrlimit_cgroup_move_task(s
  if (memrcg == old_memrcg)
   goto out;

+ /*
+  * NOTE: Even though we do the necessary checks in can_attach(),
+  * by the time we come here, there is a chance that we still
+  * fail (the memrlimit cgroup has grown its usage, and the
+  * addition of total_vm will no longer fit into its limit)
+  */
  if (res_counter_charge(&memrcg->as_res, (mm->total_vm << PAGE_SHIFT)))
   goto out;
  res_counter_uncharge(&old_memrcg->as_res, (mm->total_vm << PAGE_SHIFT));
@@ -231,6 +270,7 @@ struct cgroup_subsys memrlimit_cgroup_su
  .destroy = memrlimit_cgroup_destroy,
  .populate = memrlimit_cgroup_populate,
  .attach = memrlimit_cgroup_move_task,
+ .can_attach = memrlimit_cgroup_can_move_task,
  .mm_owner_changed = memrlimit_cgroup_mm_owner_changed,
  .early_init = 0,
 };
diff -puN kernel/res_counter.c~memrlimit-cgroup-fix-attach-task kernel/res_counter.c
diff -puN include/linux/res_counter.h~memrlimit-cgroup-fix-attach-task include/linux/res_counter.h
--- linux-2.6.26-rc5/include/linux/res_counter.h~memrlimit-cgroup-fix-attach-task 2008-06-19
22:52:17.000000000 +0530
+++ linux-2.6.26-rc5-balbir/include/linux/res_counter.h 2008-06-19 23:05:05.000000000 +0530
@@ -153,4 +153,22 @@ static inline void res_counter_reset_fai
  cnt->failcnt = 0;
  spin_unlock_irqrestore(&cnt->lock, flags);
 }
+
+/*
+ * Add the value val to the resource counter and check if we are
```

```
+ * still under the limit.
+ */
+static inline bool res_counter_add_check(struct res_counter *cnt,
+      unsigned long val)
+{
+ bool ret = false;
+ unsigned long flags;
+
+ spin_lock_irqsave(&cnt->lock, flags);
+ if (cnt->usage + val < cnt->limit)
+  ret = true;
+ spin_unlock_irqrestore(&cnt->lock, flags);
+ return ret;
+}
+
 #endif
```

–

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by KAMEZAWA Hiroyuki on Fri, 20 Jun 2008 00:09:48 GMT
View Forum Message <> Reply to Message

On Thu, 19 Jun 2008 23:55:56 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> * KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> [2008-06-19 12:14:35]:
>
> > I used memrlimit cgroup at the first time.
> >
> > May I ask a question about memrlimit cgroup ?
> >
>
> Hi, Kamezawa-San,
>
> Could you please review/test the patch below to see if it solves your
> problem? If it does, I'll push it up to Andrew
>

At quick glance,
> + /*
> + * NOTE: Even though we do the necessary checks in can_attach(),
> + * by the time we come here, there is a chance that we still
> + * fail (the memrlimit cgroup has grown its usage, and the
> + * addition of total_vm will no longer fit into its limit)
> + */
I don't like this kind of holes. Considering tests which are usually done
by developpers, the problem seems not to be mentioned as "rare"..
It seems we can easily cause Warning. right ?

Even if you don't want to handle this case now, please mention as "TBD"
rather than as "NOTE".

> +
> +/*
> + * Add the value val to the resource counter and check if we are
> + * still under the limit.
> + */
> +static inline bool res_counter_add_check(struct res_counter *cnt,
> +   unsigned long val)
> +{
> + bool ret = false;
> + unsigned long flags;
> +
> + spin_lock_irqsave(&cnt->lock, flags);
> + if (cnt->usage + val < cnt->limit)
> + ret = true;
cnt->usage + val <= cnt->limit.

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Question : memrlimit cgroup's task_move (2.6.26-rc5-mm3)
Posted by Balbir Singh on Fri, 20 Jun 2008 13:33:55 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> On Thu, 19 Jun 2008 23:55:56 +0530
> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

>
>> * KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> [2008-06-19 12:14:35]:
>>
>>> I used memrlimit cgroup at the first time.
>>>
>>> May I ask a question about memrlimit cgroup ?
>>>
>> Hi, Kamezawa-San,
>>
>> Could you please review/test the patch below to see if it solves your
>> problem? If it does, I'll push it up to Andrew
>>
>
> At quick glance,
>> + /*
>> + * NOTE: Even though we do the necessary checks in can_attach(),
>> + * by the time we come here, there is a chance that we still
>> + * fail (the memrlimit cgroup has grown its usage, and the
>> + * addition of total_vm will no longer fit into its limit)
>> + */
> I don't like this kind of holes. Considering tests which are usually done
> by developpers, the problem seems not to be mentioned as "rare"..
> It seems we can easily cause Warning. right ?
>
> Even if you don't want to handle this case now, please mention as "TBD"
> rather than as "NOTE".
>

Honestly to fix this problem completely, we need transactional management in cgroups. Both can_attach() and attach() are called with cgroup_mutex held, but total_vm is changed with mmap_sem held.

What we can do is

1. Implement a routine attach_failed() in cgroups, that is called for each task for which can_attach() succeeded, if any of the can_attach() routine returns an error
2. Do the migration in can_attach() and unroll in attach_failed()

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

_____
Containers mailing list