

---

Subject: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 25 Sep 2007 10:39:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

just for a RFC.

When I use memory controller, I notice that memory.limit\_in\_bytes shows just very big number, if unlimited.

A user(or tool) has to know that the big number(LLONG\_MAX) means "unlimited". IMHO, some interface which allows users to specify "unlimited" value is helpful.

This patch tries to define value RES\_COUNTER\_UNLIMITED (== LLONG\_MAX) and modifies an interface to support "unlimited" value.

Because this patch breaks limit\_in\_bytes to some extent, I'm glad if someone has a better idea to show unlimited value. (if some easy value means "unlimited", it's helpful. LLONG\_MAX is not easy to be recognized.)

==after this patch ==

```
[root@aworks kamezawa]# echo -n 400000000 > /opt/cgroup/memory.limit_in_bytes
[root@aworks kamezawa]# cat /opt/cgroup/memory.limit_in_bytes
400003072
[root@aworks kamezawa]# echo -n unlimited > /opt/cgroup/memory.limit_in_bytes
[root@aworks kamezawa]# cat /opt/cgroup/memory.limit_in_bytes
unlimited
```

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

```
include/linux/res_counter.h | 1 +
kernel/res_counter.c       | 11 ++++++----
2 files changed, 9 insertions(+), 3 deletions(-)
```

Index: linux-2.6.23-rc8-mm1/include/linux/res\_counter.h

```
=====
--- linux-2.6.23-rc8-mm1.orig/include/linux/res_counter.h
+++ linux-2.6.23-rc8-mm1/include/linux/res_counter.h
@@ -28,6 +28,7 @@ struct res_counter {
 * the limit that usage cannot exceed
 */
 unsigned long long limit;
+#define RES_COUNTER_UNLIMITED ((unsigned long long)LLONG_MAX)
/*
 * the number of unsuccessful attempts to consume the resource
 */
```

Index: linux-2.6.23-rc8-mm1/kernel/res\_counter.c

=====

--- linux-2.6.23-rc8-mm1.orig/kernel/res\_counter.c

+++ linux-2.6.23-rc8-mm1/kernel/res\_counter.c

@@ -16,7 +16,7 @@

```
void res_counter_init(struct res_counter *counter)
{
    spin_lock_init(&counter->lock);
- counter->limit = (unsigned long long)LLONG_MAX;
+ counter->limit = RES_COUNTER_UNLIMITED;
}
```

int res\_counter\_charge\_locked(struct res\_counter \*counter, unsigned long val)

@@ -84,7 +84,9 @@ ssize\_t res\_counter\_read(struct res\_coun

```
    s = buf;
    val = res_counter_member(counter, member);
- if (read_strategy)
+ if (*val == RES_COUNTER_UNLIMITED) {
+ s += sprintf(s, "unlimited\n", *val);
+ } else if (read_strategy)
+ s += read_strategy(*val, s);
    else
+ s += sprintf(s, "%llu\n", *val);
@@ -112,7 +114,10 @@ ssize_t res_counter_write(struct res_cou
```

```
    ret = -EINVAL;
```

```
- if (write_strategy) {
+ if ((strcmp(buf, "-1") == 0) ||
+     (strcmp(buf, "unlimited") == 0)) {
+ tmp = RES_COUNTER_UNLIMITED;
+ } else if (write_strategy) {
+     if (write_strategy(buf, &tmp)) {
+         goto out_free;
+     }
}
```

---

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.

Posted by [Balbir Singh](#) on Tue, 25 Sep 2007 10:49:18 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

KAMEZAWA Hiroyuki wrote:

> just for a RFC.  
 >  
 > When I use memory controller, I notice that memory.limit\_in\_bytes shows  
 > just very big number, if unlimited.  
 >  
 > A user(or tool) has to know that the big number(LLONG\_MAX) means "unlimited".  
 > IMHO, some interface which allows users to specify "unlimited" value is helpful.  
 >  
 > This patch tries to define value RES\_COUTNER\_UNLIMITED (== LLONG\_MAX) and  
 > modifies an interface to support "unlimited" value.  
 >  
 > Because this patch breaks limit\_in\_bytes to some extent,  
 > I'm glad if someone has a better idea to show unlimited value.  
 > (if some easy value means "unlimited", it's helpful. LLONG\_MAX is not easy  
 > to be recognized.)  
 >  
 > ==after this patch ==  
 > [root@aworks kamezawa]# echo -n 400000000 > /opt/cgroup/memory.limit\_in\_bytes  
 > [root@aworks kamezawa]# cat /opt/cgroup/memory.limit\_in\_bytes  
 > 400003072  
 > [root@aworks kamezawa]# echo -n unlimited > /opt/cgroup/memory.limit\_in\_bytes  
 > [root@aworks kamezawa]# cat /opt/cgroup/memory.limit\_in\_bytes  
 > unlimited  
 >

Hi, Kamezawa-San,

Your changes make sense, but not CLUI (Command Line Usage) sense.

1. The problem is that when we mix strings with numbers, tools that parse/use get confused and complicated
2. ULONGLONG\_MAX is a real limit, there is no such thing as unlimited.  
 If the user does ever go beyond ULONGLONG\_MAX, we will limit him :-)

Having said that, I do wish to have a more intuitive interface for users. May be a perl/python script to hide away the numbers game from the users. What do you think?

--

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: [RFC][PATCH] allow "unlimited" limit value.

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 25 Sep 2007 11:29:54 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, 25 Sep 2007 16:19:18 +0530

Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> Hi, Kamezawa-San,

>

Hi,

> Your changes make sense, but not CLUI (Command Line Usage) sense.

> 1. The problem is that when we mix strings with numbers, tools that

> parse/use get confused and complicated

yes, maybe.

> 2. ULONGLONG\_MAX is a real limit, there is no such thing as unlimited.

> If the user does ever go beyond ULONGLONG\_MAX, we will limit him :-)

>

Oh. res\_counter.c uses LONGLONG\_MAX as default value.

need fix ? or intended ?

And okay there is no "unlimited" state.

> Having said that, I do wish to have a more intuitive interface for

> users. May be a perl/python script to hide away the numbers game

> from the users. What do you think?

>

I agree with you that perl/python script can hide details. but they need knowledge about the maximum value, which is given as default value.

In short, what I want is some value like RLIM\_INFINITY in ulimit.

Because it seems that res\_counter.c will be used for other resource control purpose, I thought some generic way (value) to know/specify "the maximum value" is helpful for all resource controller interface.

If there is an consensus that treating ULONGLONG\_MAX as default, it's ok.

Thanks,

-Kame

---

Containers mailing list

[Containers@lists.linux-foundation.org](mailto:Containers@lists.linux-foundation.org)

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Balbir Singh](#) on Tue, 25 Sep 2007 11:54:38 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

KAMEZAWA Hiroyuki wrote:

> On Tue, 25 Sep 2007 16:19:18 +0530  
> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:  
>  
>> Hi, Kamezawa-San,  
>>  
> Hi,  
>  
>> Your changes make sense, but not CLUI (Command Line Usage) sense.  
>> 1. The problem is that when we mix strings with numbers, tools that  
>> parse/use get confused and complicated  
> yes, maybe.  
>  
>> 2. ULONGLONG\_MAX is a real limit, there is no such thing as unlimited.  
>> If the user does ever go beyond ULONGLONG\_MAX, we will limit him :-)  
>>  
> Oh. res\_counter.c uses LONGLONG\_MAX as default value.  
> need fix ? or intended ?

Pavel do you remember why LONG was chosen instead of ULONG?

> And okay there is no "unlimited" state.  
>  
>> Having said that, I do wish to have a more intuitive interface for  
>> users. May be a perl/python script to hide away the numbers game  
>> from the users. What do you think?  
>>  
> I agree with you that perl/python script can hide details. but they need knowledge  
> about the maximum value, which is given as default value.  
>  
> In short, what I want is some value like RLIM\_INFINITY in ulimit.  
>

I like the idea of RLIM\_INFINITY and how ulimit as a tool shows a value. I guess we need something like RES\_COUNTER\_LIMIT\_MAX and the user tool can show the limit as maximum. We could also define a special number, RES\_COUNTER\_LIMIT\_INFINITY, such that containers will not enforce limits when the limit is set to this value.

>  
> Because it seems that res\_counter.c will be used for other resource control purpose,  
> I thought some generic way (value) to know/specify "the maximum value" is helpful for  
> all resource controller interface.  
>

> If there is an consensus that treaing ULONGLONG\_MAX as default, it's ok.  
>

When I worked on the first version of res\_counters, I used 0 to indicate unlimited. When Pavel posted his version, I think derived from beancounters, we did not want to have unlimited containers, so he used the maximum value

> Thanks,  
> -Kame  
>

Thanks for looking into this,

--

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: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Pavel Emelianov](#) on Tue, 25 Sep 2007 13:06:22 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Balbir Singh wrote:

> KAMEZAWA Hiroyuki wrote:

>> On Tue, 25 Sep 2007 16:19:18 +0530

>> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

>>

>>> Hi, Kamezawa-San,

>>>

>> Hi,

>>

>>> Your changes make sense, but not CLUI (Command Line Usage) sense.

>>> 1. The problem is that when we mix strings with numbers, tools that

>>> parse/use get confused and complicated

>> yes, maybe.

>>

>>> 2. ULONGLONG\_MAX is a real limit, there is no such thing as unlimited.

>>> If the user does ever go beyond ULONGLONG\_MAX, we will limit him :-)

>>>

>> Oh. res\_counter.c uses LONGLONG\_MAX as default value.

>> need fix ? or intended ?

>  
> Pavel do you remember why LONG was chosen instead of ULONG?

To prevent the overflow in "charge" routine.  
See, if you add two numbers less than LONG\_MAX, but with  
unsigned long type each, you will never have an overflowed result.

>> And okay there is no "unlimited" state.  
>>  
>>> Having said that, I do wish to have a more intuitive interface for  
>>> users. May be a perl/python script to hide away the numbers game  
>>> from the users. What do you think?  
>>>  
>> I agree with you that perl/python script can hide details. but they need knowledge  
>> about the maximum value, which is given as default value.  
>>  
>> In short, what I want is some value like RLIM\_INFINITY in ulimit.  
>>  
>  
> I like the idea of RLIM\_INFINITY and how ulimit as a tool shows  
> a value. I guess we need something like RES\_COUNTER\_LIMIT\_MAX  
> and the user tool can show the limit as maximum. We could also  
> define a special number, RES\_COUNTER\_LIMIT\_INFINITY, such that  
> containers will not enforce limits when the limit is set to  
> this value.  
>  
>> Because it seems that res\_counter.c will be used for other resource control purpose,  
>> I thought some generic way (value) to know/specify "the maximum value" is helpful for  
>> all resource controller interface.  
>>  
>> If there is an consensus that treating ULONGLONG\_MAX as default, it's ok.  
>>  
>  
> When I worked on the first version of res\_counters, I used 0 to indicate  
> unlimited. When Pavel posted his version, I think derived from  
> beancounters, we did not want to have unlimited containers, so he used  
> the maximum value

Yup! Setting LONGMAX pages for container means that this container  
is unlimited, since there're no such many pages on any arch :)

>> Thanks,  
>> -Kame  
>>  
>  
> Thanks for looking into this,  
>

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Balbir Singh](#) on Tue, 25 Sep 2007 13:31:54 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Pavel Emelyanov wrote:

> Balbir Singh wrote:

>> KAMEZAWA Hiroyuki wrote:

>>> On Tue, 25 Sep 2007 16:19:18 +0530

>>> Balbir Singh <[balbir@linux.vnet.ibm.com](mailto:balbir@linux.vnet.ibm.com)> wrote:

>>>

>>>> Hi, Kamezawa-San,

>>>>

>>> Hi,

>>>

>>>> Your changes make sense, but not CLUI (Command Line Usage) sense.

>>>> 1. The problem is that when we mix strings with numbers, tools that

>>>> parse/use get confused and complicated

>>> yes, maybe.

>>>

>>>> 2. ULONGLONG\_MAX is a real limit, there is no such thing as unlimited.

>>>> If the user does ever go beyond ULONGLONG\_MAX, we will limit him :-)

>>>>

>>> Oh. res\_counter.c uses LONGLONG\_MAX as default value.

>>> need fix ? or intended ?

>> Pavel do you remember why LONG was chosen instead of ULONG?

>

> To prevent the overflow in "charge" routine.

> See, if you add two numbers less than LONG\_MAX, but with

> unsigned long type each, you will never have an overflowed result.

>

Aah.. Thanks, my memory short circuited on me.

>>> And okay there is no "unlimited" state.

>>>

>>>> Having said that, I do wish to have a more intuitive interface for

>>>> users. May be a perl/python script to hide away the numbers game

>>>> from the users. What do you think?

>>>>

>>> I agree with you that perl/python script can hide details. but they need knowledge

>>> about the maximum value, which is given as default value.

>>>



>>> In short, what I want is some value like RLIM\_INFINITY in ulimit.  
>>>  
>> I like the idea of RLIM\_INFINITY and how ulimit as a tool shows  
>> a value. I guess we need something like RES\_COUNTER\_LIMIT\_MAX  
>> and the user tool can show the limit as maximum. We could also  
>> define a special number, RES\_COUNTER\_LIMIT\_INFINITY, such that  
>> containers will not enforce limits when the limit is set to  
>> this value.  
>>  
>>> Because it seems that res\_counter.c will be used for other resource control purpose,  
>>> I thought some generic way (value) to know/specify "the maximum value" is helpful for  
>>> all resource controller interface.  
>>>  
>>> If there is an consensus that treating ULONGLONG\_MAX as default, it's ok.  
>>>  
>> When I worked on the first version of res\_counters, I used 0 to indicate  
>> unlimited. When Pavel posted his version, I think derived from  
>> beancounters, we did not want to have unlimited containers, so he used  
>> the maximum value  
>  
> Yup! Setting LONGMAX pages for container means that this container  
> is unlimited, since there're no such many pages on any arch :)  
>

Pavel, we no longer account in pages, we do it in bytes. Second,  
this assumption cannot hold for long, memory sizes are growing,  
I think we need a special value.

>>> Thanks,  
>>> -Kame  
>>>  
>> Thanks for looking into this,  
>>  
>

--

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: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Pavel Emelianov](#) on Tue, 25 Sep 2007 13:34:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Balbir Singh wrote:

> Pavel Emelyanov wrote:

>> Balbir Singh wrote:

>>> KAMEZAWA Hiroyuki wrote:

>>>> On Tue, 25 Sep 2007 16:19:18 +0530

>>>> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

>>>>

>>>>> Hi, Kamezawa-San,

>>>>>

>>>> Hi,

>>>>

>>>>> Your changes make sense, but not CLUI (Command Line Usage) sense.

>>>>> 1. The problem is that when we mix strings with numbers, tools that

>>>>> parse/use get confused and complicated

>>>> yes, maybe.

>>>>

>>>>> 2. ULONGLONG\_MAX is a real limit, there is no such thing as unlimited.

>>>>> If the user does ever go beyond ULONGLONG\_MAX, we will limit him :-)

>>>>>

>>>> Oh. res\_counter.c uses LONGLONG\_MAX as default value.

>>>> need fix ? or intended ?

>>> Pavel do you remember why LONG was chosen instead of ULONG?

>> To prevent the overflow in "charge" routine.

>> See, if you add two numbers less than LONG\_MAX, but with

>> unsigned long type each, you will never have an overflowed result.

>>

>

> Aah.. Thanks, my memory short circuited on me.

>

>>>> And okay there is no "unlimited" state.

>>>>

>>>>> Having said that, I do wish to have a more intuitive interface for

>>>>> users. May be a perl/python script to hide away the numbers game

>>>>> from the users. What do you think?

>>>>>

>>>> I agree with you that perl/python script can hide details. but they need knowledge

>>>> about the maximum value, which is given as default value.

>>>>

>>>> In short, what I want is some value like RLIM\_INFINITY in ulimit.

>>>>

>>> I like the idea of RLIM\_INFINITY and how ulimit as a tool shows

>>> a value. I guess we need something like RES\_COUNTER\_LIMIT\_MAX

>>> and the user tool can show the limit as maximum. We could also

>>> define a special number, RES\_COUNTER\_LIMIT\_INFINITY, such that

>>> containers will not enforce limits when the limit is set to

>>> this value.  
>>>  
>>>> Because it seems that res\_counter.c will be used for other resource control purpose,  
>>>> I thought some generic way (value) to know/specify "the maximum value" is helpful for  
>>>> all resource controller interface.  
>>>>  
>>>> If there is an consensus that treating ULONGLONG\_MAX as default, it's ok.  
>>>>  
>>> When I worked on the first version of res\_counters, I used 0 to indicate  
>>> unlimited. When Pavel posted his version, I think derived from  
>>> beancounters, we did not want to have unlimited containers, so he used  
>>> the maximum value  
>> Yup! Setting LONGMAX pages for container means that this container  
>> is unlimited, since there're no such many pages on any arch :)  
>>  
>  
> Pavel, we no longer account in pages, we do it in bytes. Second,  
> this assumption cannot hold for long, memory sizes are growing,  
> I think we need a special value.

I see. And I also see that we've switched into unsigned long long.

Well, no container may have the ULLMAX (or what is it?) bytes  
touched/allocated :) So I don't see any need in a special value.

>  
>>>> Thanks,  
>>>> -Kame  
>>>>  
>>> Thanks for looking into this,  
>>>  
>  
>

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 25 Sep 2007 15:05:10 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, 25 Sep 2007 17:34:00 +0400  
Pavel Emelyanov <xemul@openvz.org> wrote:  
> Well, no container may have the ULLMAX (or what is it?) bytes  
> touched/allocated :) So I don't see any need in a special value.

>  
Then, ULLMAX is default value of "not configured cgroup-resource-counter".

For make things clear for people(including not-hacker-users),  
can we have some definition as following ?

--  
#define RES\_COUNTER\_INFINITY (~0ULL)  
or some nice name

Thanks,  
-Kame

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Pavel Emelianov](#) on Tue, 25 Sep 2007 15:14:53 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

KAMEZAWA Hiroyuki wrote:

> On Tue, 25 Sep 2007 17:34:00 +0400  
> Pavel Emelianov <xemul@openvz.org> wrote:  
>> Well, no container may have the ULLMAX (or what is it?) bytes  
>> touched/allocated :) So I don't see any need in a special value.  
>>  
> Then, ULLMAX is default value of "not configured cgroup-resource-counter".  
>  
> For make things clear for people(including not-hacker-users),  
> can we have some definition as following ?  
> --  
> #define RES\_COUNTER\_INFINITY (~0ULL)  
> or some nice name

Why do we need this at all? We can simply push -1 there and be happy.

> Thanks,  
> -Kame  
>

---

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

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 25 Sep 2007 15:30:56 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, 25 Sep 2007 19:14:53 +0400  
Pavel Emelyanov <xemul@openvz.org> wrote:

```
> KAMEZAWA Hiroyuki wrote:
> > On Tue, 25 Sep 2007 17:34:00 +0400
> > Pavel Emelyanov <xemul@openvz.org> wrote:
> >> Well, no container may have the ULLMAX (or what is it?) bytes
> >> touched/allocated :) So I don't see any need in a special value.
> >>
> > Then, ULLMAX is default value of "not configured cgroup-resource-counter".
> >
> > For make things clear for people(including not-hacker-users),
> > can we have some definition as following ?
> > --
> > #define RES_COUNTER_INFINITY (~0ULL)
> > or some nice name
>
> Why do we need this at all? We can simply push -1 there and be happy.
>
Hm, can this work now ?
==
echo -1 > /cgroup/memory.limit_in_bytes
==
Or users have to do following for unlimit resource ?
==
echo some-very-very-big-number > /cgroup/memory.limit_in_bytes
```

I just think when some special value "-1" has a nice nick name, users will be happy. If I'm a novice user, I don't imagine I can write -1 to limit value. (but ok, tools can hide it for them.)

Thanks,  
-Kame

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [David Rientjes](#) on Tue, 25 Sep 2007 19:07:58 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 26 Sep 2007, KAMEZAWA Hiroyuki wrote:

```
> > > #define RES_COUNTER_INFINITY (~0ULL)
> > > or some nice name
> >
> > Why do we need this at all? We can simply push -1 there and be happy.
> >
> > Hm, can this work now ?
> ==
> echo -1 > /cgroup/memory.limit_in_bytes
> ==
> Or users have to do following for unlimit resource ?
> ==
> echo some-very-very-big-number > /cgroup/memory.limit_in_bytes
>
>
> I just think when some special value "-1" has a nice nick name, users will
> be happy. If I'm a novice user, I don't imagine I can write -1 to limit value.
> (but ok, tools can hide it for them.)
>
```

Please simply use 0 to denote unconstrained memory, it's quite obvious that nobody will sanely attach tasks to a cgroup that has no bytes of memory allowed.

In fact, I proposed this in a patch on August 27.

I really don't like the use of ULONG\_MAX to denote the absence of any memory controls for a particular container. I think 0 would be suitable since its use doesn't make any logical sense (you're not going to be assigning a set of tasks to a resource void of pages).

Signed-off-by: David Rientjes <rientjes@google.com>

```
---
Documentation/controllers/memory.txt | 5 ++++-
kernel/res_counter.c                  | 7 ++++++-
2 files changed, 9 insertions(+), 3 deletions(-)
```

```
diff --git a/Documentation/controllers/memory.txt b/Documentation/controllers/memory.txt
--- a/Documentation/controllers/memory.txt
+++ b/Documentation/controllers/memory.txt
@@ -164,13 +164,16 @@ c. Enable CONFIG_CONTAINER_MEM_CONT
# echo $$ > /containers/0/tasks
```

Since now we're in the 0 container,  
-We can alter the memory limit:

+We can alter the memory limit (in pages):  
# echo -n 6000 > /containers/0/memory.limit

We can check the usage:  
# cat /containers/0/memory.usage  
25

+If memory.limit is set to 0, no charge is accumulated for that resource  
+controller.

+  
The memory.failcnt field gives the number of times that the container limit was exceeded.

```
diff --git a/kernel/res_counter.c b/kernel/res_counter.c
--- a/kernel/res_counter.c
+++ b/kernel/res_counter.c
@@ -16,12 +16,15 @@
void res_counter_init(struct res_counter *counter)
{
    spin_lock_init(&counter->lock);
- counter->limit = (unsigned long)LONG_MAX;
}

int res_counter_charge_locked(struct res_counter *counter, unsigned long val)
{
- if (counter->usage + val > counter->limit) {
+ /*
+  * If 'memory.limit' is set to 0, there is no charge to this
+  * res_counter.
+  */
+ if (counter->limit && counter->usage + val > counter->limit) {
    counter->failcnt++;
    return -ENOMEM;
}
```

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Balbir Singh](#) on Tue, 25 Sep 2007 19:21:59 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

David Rientjes wrote:  
> On Wed, 26 Sep 2007, KAMEZAWA Hiroyuki wrote:  
>  
>>>> #define RES\_COUNTER\_INFINITY (~0ULL)

```

>>>> or some nice name
>>> Why do we need this at all? We can simply push -1 there and be happy.
>>>
>> Hm, can this work now ?
>> ==
>> echo -1 > /cgroup/memory.limit_in_bytes
>> ==
>> Or users have to do following for unlimit resource ?
>> ==
>> echo some-very-very-big-number > /cgroup/memory.limit_in_bytes
>>
>>
>> I just think when some special value "-1" has a nice nick name, users will
>> be happy. If I'm a novice user, I don't imagine I can write -1 to limit value.
>> (but ok, tools can hide it for them.)
>>
>
> Please simply use 0 to denote unconstrained memory, it's quite obvious
> that nobody will sanely attach tasks to a cgroup that has no bytes of
> memory allowed.
>

```

Yes, I prefer 0 as well and had that in a series in the Lost World of my earlier memory/RSS controller patches. I feel now that 0 is a bit confusing, we don't use 0 to mean unlimited, unless we treat the memory.limit\_in\_bytes value as boolean. 0 is false, meaning there is no limit, > 0 is true, which means the limit is set and the value is specified to the value read out.

```

> diff --git a/kernel/res_counter.c b/kernel/res_counter.c
> --- a/kernel/res_counter.c
> +++ b/kernel/res_counter.c
> @@ -16,12 +16,15 @@
> void res_counter_init(struct res_counter *counter)
> {
>     spin_lock_init(&counter->lock);
>     counter->limit = (unsigned long)LONG_MAX;

```

So, we create all containers with infinite limit?

```

> }
>
> int res_counter_charge_locked(struct res_counter *counter, unsigned long val)
> {
>     if (counter->usage + val > counter->limit) {
> + /*
> +  * If 'memory.limit' is set to 0, there is no charge to this

```



nit pick, should be memory.limit\_in\_bytes

```
> + * res_counter.  
> + */  
> + if (counter->limit && counter->usage + val > counter->limit) {  
>   counter->failcnt++;  
>   return -ENOMEM;  
> }
```

--

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: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [David Rientjes](#) on Tue, 25 Sep 2007 19:30:39 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 26 Sep 2007, Balbir Singh wrote:

```
> Yes, I prefer 0 as well and had that in a series in the Lost World  
> of my earlier memory/RSS controller patches. I feel now that 0 is  
> a bit confusing, we don't use 0 to mean unlimited, unless we  
> treat the memory.limit_in_bytes value as boolean. 0 is false,  
> meaning there is no limit, > 0 is true, which means the limit  
> is set and the value is specified to the value read out.  
>
```

I think any user who attaches a task that is still running to cgroup that has memory.limit\_in\_bytes specified as 0 will realize quickly that it's not doing anything to limit memory. 0 is the best choice for denoting unlimited memory limits.

```
> > diff --git a/kernel/res_counter.c b/kernel/res_counter.c  
> > --- a/kernel/res_counter.c  
> > +++ b/kernel/res_counter.c  
> > @@ -16,12 +16,15 @@  
> > void res_counter_init(struct res_counter *counter)  
> > {  
> >   spin_lock_init(&counter->lock);  
> >   counter->limit = (unsigned long)LONG_MAX;
```

>  
> So, we create all containers with infinite limit?  
>

Yeah, since you kcalloc'd the struct mem\_cgroup, the struct res\_counter will also be zero'd and inherently have a limit of 0. It's far better than any arbitrary value you're going to give them, unless they inherit the value of their parent.

```
> > }  
> >  
> > int res_counter_charge_locked(struct res_counter *counter, unsigned long val)  
> > {  
> > - if (counter->usage + val > counter->limit) {  
> > + /*  
> > +  * If 'memory.limit' is set to 0, there is no charge to this  
>  
> nit pick, should be memory.limit_in_bytes  
>
```

This is from a month ago, I'm assuming more has changed than just the name here :)

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Paul Menage](#) on Tue, 25 Sep 2007 19:35:16 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 9/25/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

```
>  
> nit pick, should be memory.limit_in_bytes  
>
```

Can we reconsider this? I do think that plain "limit" would enable you to have a more consistent API across all resource counters users.

I realise that there's the issue that if someone's never heard of the memory controller and never read any docs for it then they might be very briefly uncertain as to the units, but really, that's what documentation is for.

Paul

---

Containers mailing list

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [David Rientjes](#) on Tue, 25 Sep 2007 19:40:38 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, 25 Sep 2007, Paul Menage wrote:

> > nit pick, should be memory.limit\_in\_bytes  
> >  
>  
> Can we reconsider this? I do think that plain "limit" would enable you  
> to have a more consistent API across all resource counters users.  
>

Why aren't limits expressed in kilobytes? All architectures have  
PAGE\_SIZE defined on that order.

If I echo -n 8191 > memory.limit\_in\_bytes, I'm still only going to be able  
to charge one page on my x86\_64. And then my program's malloc(5000) is  
going to fail, which leads to the inevitable head scratching.

I think it would be best to express memory.limit in terms of KB, divide  
that by PAGE\_SIZE to store internally in res\_counter.limit, deal with  
charging for memory internally in terms of number of pages, and exposing  
it back to userspace in terms of res\_counter.limit \* PAGE\_SIZE (KB).

David

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Paul Menage](#) on Tue, 25 Sep 2007 20:00:34 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 9/25/07, David Rientjes <[rientjes@google.com](mailto:rientjes@google.com)> wrote:

>  
> If I echo -n 8191 > memory.limit\_in\_bytes, I'm still only going to be able  
> to charge one page on my x86\_64. And then my program's malloc(5000) is  
> going to fail, which leads to the inevitable head scratching.

This is a very unrealistic argument. Page-size rounding really has no

effect on any reasonable-sized memory cgroup.

Expressing it in bytes seems reasonable to me, since they are after all the fundamental unit that's being counted ("kilobytes" are explicitly an aggregation of "bytes").

Paul

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [David Rientjes](#) on Tue, 25 Sep 2007 20:07:56 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, 25 Sep 2007, Paul Menage wrote:

> > If I echo -n 8191 > memory.limit\_in\_bytes, I'm still only going to be able  
> > to charge one page on my x86\_64. And then my program's malloc(5000) is  
> > going to fail, which leads to the inevitable head scratching.  
>  
> This is a very unrealistic argument. Page-size rounding really has no  
> effect on any reasonable-sized memory cgroup.  
>

It doesn't matter. When I cat my cgroup's memory.limit (or memory.limit\_in\_bytes), I should see the total number of bytes that my applications are allowed. That's not an unrealistic expectation of a system that is expressly designed to control my memory. I don't want to see a value that is close to what I'm allowed, thanks.

Storing it internally as the number of pages makes the implementation simpler since memory controls are only imposed on pages anyway and you get the added bonus of integer division truncating in C so that when you cat the file it will display the correct number of bytes modulo PAGE\_SIZE.

> Expressing it in bytes seems reasonable to me, since they are after  
> all the fundamental unit that's being counted ("kilobytes" are  
> explicitly an aggregation of "bytes").  
>

That fundamental unit being charged are pages, so any memory limit that has a finer granularity than kilobytes is just plain wrong.

David

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Paul Menage](#) on Tue, 25 Sep 2007 20:12:40 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 9/25/07, David Rientjes <[rientjes@google.com](mailto:rientjes@google.com)> wrote:  
> It doesn't matter. When I cat my cgroup's memory.limit (or  
> memory.limit\_in\_bytes), I should see the total number of bytes that my  
> applications are allowed. That's not an unrealistic expectation of a  
> system that is expressly designed to control my memory. I don't want to  
> see a value that is close to what I'm allowed, thanks.

So round up to the nearest page. Then you'll get what you asked for so  
you can't get broken by the rounding.

>  
> Storing it internally as the number of pages makes the implementation  
> simpler since memory controls are only imposed on pages anyway and you get  
> the added bonus of integer division truncating in C so that when you cat  
> the file it will display the correct number of bytes modulo PAGE\_SIZE.

Storing it internally as a number of pages is fine. I'm more concerned  
about the userspace API, since that will be very hard to change.

>  
> That fundamental unit being charged are pages,

No, that just happens to be the implementation mechanism in this controller.

You could be internally charging in much larger units (e.g. fake numa  
nodes) or smaller units (slab object allocations).

Paul

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [David Rientjes](#) on Tue, 25 Sep 2007 20:32:52 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, 25 Sep 2007, Paul Menage wrote:

> > It doesn't matter. When I cat my cgroup's memory.limit (or  
> > memory.limit\_in\_bytes), I should see the total number of bytes that my  
> > applications are allowed. That's not an unrealistic expectation of a  
> > system that is expressly designed to control my memory. I don't want to  
> > see a value that is close to what I'm allowed, thanks.  
>  
> So round up to the nearest page. Then you'll get what you asked for so  
> you can't get broken by the rounding.  
>

If you're fine with rounding up to the nearest page, then what's the point of exposing it as a number of bytes?? You'll never get a granularity finer than a kilobyte.

So by expressing it in terms of bytes instead of kilobytes, you're just making the largest amount of memory allowed via this interface smaller that it should have to be. That is absolutely horrid in terms of scalability and you're never going to be able to get rid of it because everything that interfaces with it by then will have been written in terms of bytes.

> > That fundamental unit being charged are pages,  
>  
> No, that just happens to be the implementation mechanism in this controller.  
>

And this controller owns the memory.limit file so it can express its memory limits in whatever unit it wants.

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Paul Menage](#) on Tue, 25 Sep 2007 20:40:52 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 9/25/07, David Rientjes <[rientjes@google.com](mailto:rientjes@google.com)> wrote:

>  
> If you're fine with rounding up to the nearest page, then what's the point  
> of exposing it as a number of bytes?? You'll never get a granularity  
> finer than a kilobyte.

API != implementation.

>  
> So by expressing it in terms of bytes instead of kilobytes, you're just  
> making the largest amount of memory allowed via this interface smaller  
> that is should have to be.

Yes, that's true. With a 64-bit count in bytes, we can only limit people to 16 exabytes of memory. We're going to bump up against that limit in no time.

>  
> > That fundamental unit being charged are pages,  
> >  
> > No, that just happens to be the implementation mechanism in this controller.  
> >  
>  
> And this controller owns the memory.limit file so it can express its  
> memory limits in whatever unit it wants.  
>

Right, but it would be nice to have different memory controllers be API-compatible with one another. Bytes is the lowest common denominator.

Paul

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [David Rientjes](#) on Tue, 25 Sep 2007 20:58:29 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, 25 Sep 2007, Paul Menage wrote:

> > If you're fine with rounding up to the nearest page, then what's the point  
> > of exposing it as a number of bytes?? You'll never get a granularity  
> > finer than a kilobyte.  
>  
> API != implementation.  
>

Having the limit expressed and configurable in bytes suggests that it can be defined on that granularity which is obviously wrong.

> > So by expressing it in terms of bytes instead of kilobytes, you're just  
> > making the largest amount of memory allowed via this interface smaller

> > that is should have to be.  
>  
> Yes, that's true. With a 64-bit count in bytes, we can only limit  
> people to 16 exabytes of memory. We're going to bump up against that  
> limit in no time.  
>

So, by your logic, it would be fine to express it in bits too.

> > And this controller owns the memory.limit file so it can express its  
> > memory limits in whatever unit it wants.  
> >  
>  
> Right, but it would be nice to have different memory controllers be  
> API-compatible with one another. Bytes is the lowest common  
> denominator.  
>

Please cite examples of other memory controllers that you can imagine would actually support (not expose to userspace, but support) memory limits in terms of anything smaller than kilobytes and how you plan on charging for that memory as a fraction of a page size and that has any reasonable hope of ever being efficient.

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Paul Menage](#) on Wed, 26 Sep 2007 00:06:21 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 9/25/07, David Rientjes <[rientjes@google.com](mailto:rientjes@google.com)> wrote:

>  
> Having the limit expressed and configurable in bytes suggests that it can  
> be defined on that granularity which is obviously wrong.

One of the other options suggested was that you can write a value in bytes, and the value you can read back from there will reflect the real limit, with any associated granularity/rounding.

>  
> > > So by expressing it in terms of bytes instead of kilobytes, you're just  
> > > making the largest amount of memory allowed via this interface smaller  
> > > that is should have to be.  
> >  
> > Yes, that's true. With a 64-bit count in bytes, we can only limit



> > people to 16 exabytes of memory. We're going to bump up against that  
> > limit in no time.  
> >  
>  
> So, by your logic, it would be fine to express it in bits too.

I don't think it would be much of a scalability limit to express it in bits, no. Of course, it would be a bit silly. Bytes are the natural counting units for memory - e.g. they're the units you get back when you call sizeof(), or you pass to malloc().

>  
> Please cite examples of other memory controllers that you can imagine  
> would actually support (not expose to userspace, but support) memory  
> limits in terms of anything smaller than kilobytes

Pavel's kernel memory controller, posted to this list this morning, appears to charge for each object based on its size in bytes.

I could also imagine that a filesystem that packs short files or tails into partial pages could charge based on those partial pages, although I don't know of any such controller.

Paul

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [KAMEZAWA Hiroyuki](#) on Wed, 26 Sep 2007 01:23:43 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 26 Sep 2007 00:51:59 +0530  
Balbir Singh <[balbir@linux.vnet.ibm.com](mailto:balbir@linux.vnet.ibm.com)> wrote:

> David Rientjes wrote:

> Yes, I prefer 0 as well and had that in a series in the Lost World  
> of my earlier memory/RSS controller patches. I feel now that 0 is  
> a bit confusing, we don't use 0 to mean unlimited, unless we  
> treat the memory.limit\_in\_bytes value as boolean. 0 is false,  
> meaning there is no limit, > 0 is true, which means the limit  
> is set and the value is specified to the value read out.

I prefer 0 than -1, too

Thanks,  
-Kame

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Pavel Emelianov](#) on Wed, 26 Sep 2007 09:45:47 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

KAMEZAWA Hiroyuki wrote:

> On Wed, 26 Sep 2007 00:51:59 +0530  
> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:  
>  
>> David Rientjes wrote:  
>  
>> Yes, I prefer 0 as well and had that in a series in the Lost World  
>> of my earlier memory/RSS controller patches. I feel now that 0 is  
>> a bit confusing, we don't use 0 to mean unlimited, unless we  
>> treat the memory.limit\_in\_bytes value as boolean. 0 is false,  
>> meaning there is no limit, > 0 is true, which means the limit  
>> is set and the value is specified to the value read out.  
>  
> I prefer 0 than -1, too

Remember, that we may use resource counters for other control groups  
0 would make ore sense, like for numfile CG. 0 can mean that this  
group is not allowed to open any files. Treating 0 as unlimited for  
some CGs and as 0 for others is a mess.

> Thanks,  
> -Kame  
>  
>

---

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

---

---

Subject: Re: [RFC][PATCH] allow "unlimited" limit value.  
Posted by [Balbir Singh](#) on Wed, 26 Sep 2007 10:59:32 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Pavel Emelyanov wrote:

> KAMEZAWA Hiroyuki wrote:

>> On Wed, 26 Sep 2007 00:51:59 +0530

>> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

>>

>>> David Rientjes wrote:

>>> Yes, I prefer 0 as well and had that in a series in the Lost World  
>>> of my earlier memory/RSS controller patches. I feel now that 0 is  
>>> a bit confusing, we don't use 0 to mean unlimited, unless we  
>>> treat the memory.limit\_in\_bytes value as boolean. 0 is false,  
>>> meaning there is no limit, > 0 is true, which means the limit  
>>> is set and the value is specified to the value read out.

>> I prefer 0 than -1, too

>

> Remember, that we may use resource counters for other control groups

> 0 would make ore sense, like for numfile CG. 0 can mean that this

> group is not allowed to open any files. Treating 0 as unlimited for

> some CGs and as 0 for others is a mess.

>

I disagree, numfile CG using 0 will not work, cause you'll not be able  
to do anything with 0, you can't even cat the numfile.limit file; for  
that matter anything with 0 will not work. You'll always exceed the  
limit.

Setting 0 to mean unlimited might make sense.

--

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: [RFC][PATCH] allow "unlimited" limit value.

Posted by [Pavel Emelianov](#) on Wed, 26 Sep 2007 11:02:31 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Balbir Singh wrote:

> Pavel Emelyanov wrote:

>> KAMEZAWA Hiroyuki wrote:

>>> On Wed, 26 Sep 2007 00:51:59 +0530

>>> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

>>>

>>>> David Rientjes wrote:  
>>>> Yes, I prefer 0 as well and had that in a series in the Lost World  
>>>> of my earlier memory/RSS controller patches. I feel now that 0 is  
>>>> a bit confusing, we don't use 0 to mean unlimited, unless we  
>>>> treat the memory.limit\_in\_bytes value as boolean. 0 is false,  
>>>> meaning there is no limit, > 0 is true, which means the limit  
>>>> is set and the value is specified to the value read out.  
>>> I prefer 0 than -1, too  
>> Remember, that we may use resource counters for other control groups  
>> 0 would make ore sense, like for numfile CG. 0 can mean that this  
>> group is not allowed to open any files. Treating 0 as unlimited for  
>> some CGs and as 0 for others is a mess.  
>>  
>  
> I disagree, numfile CG using 0 will not work, cause you'll not be able  
> to do anything with 0, you can't even cat the numfile.limit file; for

So what? I'm the administrator and I don't want this particular subgroup  
to open any files :)

> that matter anything with 0 will not work. You'll always exceed the

Yet again - I don't want some subgroup to consume any of some resource.  
E.g. I don't want some subgroup to use any private pages :) shared  
only, what can I do?

> limit.  
>  
> Setting 0 to mean unlimited might make sense.

Setting 0 as unlimited is used nowhere in the kernel, isn't it?

Thanks,  
Pavel

---

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

---