Subject: problem with ZONE_MOVABLE.
Posted by KAMEZAWA Hiroyuki on Thu, 13 Sep 2007 10:07:19 GMT
View Forum Message <> Reply to Message

Hi,

While I'm playing with memory controller of 2.6.23-rc4-mm1, I met following.

==
[root@drpq test-2.6.23-rc4-mm1]# echo $$ > /opt/mem_control/group_1/tasks
[root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
32768
[root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
286
// Memory is limited to 512 GiB. try "dd" 1GiB (page size is 16KB)

[root@drpq test-2.6.23-rc4-mm1]# dd if=/dev/zero of=/tmp/tmpfile bs=1024 count=1048576
Killed
[root@drpq test-2.6.23-rc4-mm1]# ls
Killed
//above are caused by OOM.
[root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
32763
[root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
32768
// fully filled by page cache. no reclaim run.
==

The reason this happens is because I used kernelcore= boot option, i.e
ZONE_MOVABLE. Seems try_to_free_mem_container_pages() ignores ZONE_MOVABLE.

Quick fix is attached, but Mel's one-zonelist-pernode patch may change this.
I'll continue to watch.

Thanks,
-Kame
==
Now, there is ZONE_MOVABLE...

page cache and user pages are allocated from gfp_zone(GFP_HIGHUSER_MOVABLE)

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
---
 mm/vmscan.c | 9 ++-------
 1 file changed, 2 insertions(+), 7 deletions(-)

Index: linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
===

```
--- linux-2.6.23-rc4-mm1.bak.orig/mm/vmscan.c
+++ linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
@@ -1351,12 +1351,6 @@ unsigned long try_to_free_pages(struct z

 #ifdef CONFIG_CONTAINER_MEM_CONT

-#ifdef CONFIG_HIGHMEM
-#define ZONE_USERPAGES ZONE_HIGHMEM
-#else
-#define ZONE_USERPAGES ZONE_NORMAL
-#endif
-
 unsigned long try_to_free_mem_container_pages(struct mem_container *mem_cont)
 {
  struct scan_control sc = {
@@ -1371,9 +1365,10 @@ unsigned long try_to_free_mem_container_
  };
  int node;
  struct zone **zones;
+ int target_zone = gfp_zone(GFP_HIGHUSER_MOVABLE);

  for_each_online_node(node) {
-  zones = NODE_DATA(node)->node_zonelists[ZONE_USERPAGES].zones;
+  zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
   if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
    return 1;
  }
```

Subject: Re: problem with ZONE_MOVABLE.
Posted by Balbir Singh on Thu, 13 Sep 2007 10:30:06 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> Hi,
>
> While I'm playing with memory controller of 2.6.23-rc4-mm1, I met following.
>

> ==
> [root@drpq test-2.6.23-rc4-mm1]# echo $$ > /opt/mem_control/group_1/tasks
> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
> 32768
> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
> 286
> // Memory is limited to 512 GiB. try "dd" 1GiB (page size is 16KB)
>
> [root@drpq test-2.6.23-rc4-mm1]# dd if=/dev/zero of=/tmp/tmpfile bs=1024 count=1048576
> Killed
> [root@drpq test-2.6.23-rc4-mm1]# ls
> Killed
> //above are caused by OOM.
> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
> 32763
> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
> 32768
> // fully filled by page cache. no reclaim run.
> ==
>
> The reason this happens is because I used kernelcore= boot option, i.e
> ZONE_MOVABLE. Seems try_to_free_mem_container_pages() ignores ZONE_MOVABLE.
>
> Quick fix is attached, but Mel's one-zonelist-pernode patch may change this.
> I'll continue to watch.
>
> Thanks,
> -Kame
> ==
> Now, there is ZONE_MOVABLE...
>
> page cache and user pages are allocated from gfp_zone(GFP_HIGHUSER_MOVABLE)
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> ---
> mm/vmscan.c | 9 ++-------
> 1 file changed, 2 insertions(+), 7 deletions(-)
>
> Index: linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
> ===
> --- linux-2.6.23-rc4-mm1.bak.orig/mm/vmscan.c
> +++ linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
> @@ -1351,12 +1351,6 @@ unsigned long try_to_free_pages(struct z
>
> #ifdef CONFIG_CONTAINER_MEM_CONT
>
> -#ifdef CONFIG_HIGHMEM
> -#define ZONE_USERPAGES ZONE_HIGHMEM

> -#else
> -#define ZONE_USERPAGES ZONE_NORMAL
> -#endif
> -
> unsigned long try_to_free_mem_container_pages(struct mem_container *mem_cont)
> {
>   struct scan_control sc = {
> @@ -1371,9 +1365,10 @@ unsigned long try_to_free_mem_container_
>   };
>   int node;
>   struct zone **zones;
> + int target_zone = gfp_zone(GFP_HIGHUSER_MOVABLE);
>
>   for_each_online_node(node) {
> - zones = NODE_DATA(node)->node_zonelists[ZONE_USERPAGES].zones;
> + zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
>   if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
>   return 1;
>   }

Mel, has sent out a fix (for the single zonelist) that conflicts with
this one. Your fix looks correct to me, but it will be over ridden
by Mel's fix (once those patches are in -mm).

--
 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: problem with ZONE_MOVABLE.
Posted by KAMEZAWA Hiroyuki on Thu, 13 Sep 2007 10:35:47 GMT
View Forum Message <> Reply to Message

On Thu, 13 Sep 2007 16:00:06 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

>
> Mel, has sent out a fix (for the single zonelist) that conflicts with
> this one. Your fix looks correct to me, but it will be over ridden
> by Mel's fix (once those patches are in -mm).
>
Ah yes. just for notification.

thanks,
-Kame

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

---

Subject: Re: problem with ZONE_MOVABLE.
Posted by mel on Thu, 13 Sep 2007 13:11:18 GMT
View Forum Message <> Reply to Message

On (13/09/07 19:07), KAMEZAWA Hiroyuki didst pronounce:
> Hi,
>
> While I'm playing with memory controller of 2.6.23-rc4-mm1, I met following.
>
> ==
> [root@drpq test-2.6.23-rc4-mm1]# echo $$ > /opt/mem_control/group_1/tasks
> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
> 32768
> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
> 286
> // Memory is limited to 512 GiB. try "dd" 1GiB (page size is 16KB)
>
> [root@drpq test-2.6.23-rc4-mm1]# dd if=/dev/zero of=/tmp/tmpfile bs=1024 count=1048576
> Killed
> [root@drpq test-2.6.23-rc4-mm1]# ls
> Killed
> //above are caused by OOM.
> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
> 32763
> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
> 32768
> // fully filled by page cache. no reclaim run.
> ==
>
> The reason this happens is because I used kernelcore= boot option, i.e
> ZONE_MOVABLE. Seems try_to_free_mem_container_pages() ignores ZONE_MOVABLE.
>
> Quick fix is attached, but Mel's one-zonelist-pernode patch may change this.
> I'll continue to watch.
>

You are right on both counts. This is a valid fix but
one-zonelist-pernode overwrites it. Specifically the code in question

with one-zonelist will look like;

```
 for_each_online_node(node) {
  zonelist = &NODE_DATA(node)->node_zonelist;
  if (do_try_to_free_pages(zonelist, sc.gfp_mask, &sc))
   return 1;
 }
```

We should be careful that this problem does not get forgotten about if
one-zonelist gets delayed for a long period of time. Have the fix at the
end of the container patchset where it can be easily dropped if
one-zonelist is merged.

Thanks

> Thanks,
> -Kame
> ==
> Now, there is ZONE_MOVABLE...
>
> page cache and user pages are allocated from gfp_zone(GFP_HIGHUSER_MOVABLE)
>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

Acked-by: Mel Gorman <mel@csn.ul.ie>

> ---
> mm/vmscan.c | 9 ++-------
> 1 file changed, 2 insertions(+), 7 deletions(-)
>
> Index: linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
> ===
> --- linux-2.6.23-rc4-mm1.bak.orig/mm/vmscan.c
> +++ linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
> @@ -1351,12 +1351,6 @@ unsigned long try_to_free_pages(struct z
>
> #ifdef CONFIG_CONTAINER_MEM_CONT
>
> -#ifdef CONFIG_HIGHMEM
> -#define ZONE_USERPAGES ZONE_HIGHMEM
> -#else
> -#define ZONE_USERPAGES ZONE_NORMAL
> -#endif
> -
> unsigned long try_to_free_mem_container_pages(struct mem_container *mem_cont)
> {
>   struct scan_control sc = {
> @@ -1371,9 +1365,10 @@ unsigned long try_to_free_mem_container_

> };
> int node;
> struct zone **zones;
> + int target_zone = gfp_zone(GFP_HIGHUSER_MOVABLE);
>
> for_each_online_node(node) {
> - zones = NODE_DATA(node)->node_zonelists[ZONE_USERPAGES].zones;
> + zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
> if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
> return 1;
> }
>
>
>
>
>
>
> --
> To unsubscribe, send a message with 'unsubscribe linux-mm' in
> the body to majordomo@kvack.org. For more info on Linux MM,
> see: http://www.linux-mm.org/ .
> Don't email: email@kvack.org

--
--
Mel Gorman
Part-time Phd Student Linux Technology Center
University of Limerick IBM Dublin Software Lab

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

---

Subject: Re: problem with ZONE_MOVABLE.
Posted by Balbir Singh on Thu, 13 Sep 2007 15:53:30 GMT
View Forum Message <> Reply to Message

Mel Gorman wrote:
> On (13/09/07 19:07), KAMEZAWA Hiroyuki didst pronounce:
>> Hi,
>>
>> While I'm playing with memory controller of 2.6.23-rc4-mm1, I met following.
>>
>> ==
>> [root@drpq test-2.6.23-rc4-mm1]# echo $$ > /opt/mem_control/group_1/tasks
>> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
>> 32768

>> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
>> 286
>> // Memory is limited to 512 GiB. try "dd" 1GiB (page size is 16KB)
>>
>> [root@drpq test-2.6.23-rc4-mm1]# dd if=/dev/zero of=/tmp/tmpfile bs=1024 count=1048576
>> Killed
>> [root@drpq test-2.6.23-rc4-mm1]# ls
>> Killed
>> //above are caused by OOM.
>> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
>> 32763
>> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
>> 32768
>> // fully filled by page cache. no reclaim run.
>> ==
>>
>> The reason this happens is because I used kernelcore= boot option, i.e
>> ZONE_MOVABLE. Seems try_to_free_mem_container_pages() ignores ZONE_MOVABLE.
>>
>> Quick fix is attached, but Mel's one-zonelist-pernode patch may change this.
>> I'll continue to watch.
>>
>
> You are right on both counts. This is a valid fix but
> one-zonelist-pernode overwrites it. Specifically the code in question
> with one-zonelist will look like;
>
> for_each_online_node(node) {
>   zonelist = &NODE_DATA(node)->node_zonelist;
>   if (do_try_to_free_pages(zonelist, sc.gfp_mask, &sc))
>   return 1;
> }
>
> We should be careful that this problem does not get forgotten about if
> one-zonelist gets delayed for a long period of time. Have the fix at the
> end of the container patchset where it can be easily dropped if
> one-zonelist is merged.
>
> Thanks

Yes, I second that. So, we should get KAMEZAWA's fix in.

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

_____

Subject: Re: problem with ZONE_MOVABLE.
Posted by akpm on Sat, 15 Sep 2007 00:38:35 GMT
View Forum Message <> Reply to Message

On Thu, 13 Sep 2007 16:00:06 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> KAMEZAWA Hiroyuki wrote:
> > Hi,
> >
> > While I'm playing with memory controller of 2.6.23-rc4-mm1, I met following.
> >
> > ==
> > [root@drpq test-2.6.23-rc4-mm1]# echo $$ > /opt/mem_control/group_1/tasks
> > [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
> > 32768
> > [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
> > 286
> > // Memory is limited to 512 GiB. try "dd" 1GiB (page size is 16KB)
> >
> > [root@drpq test-2.6.23-rc4-mm1]# dd if=/dev/zero of=/tmp/tmpfile bs=1024 count=1048576
> > Killed
> > [root@drpq test-2.6.23-rc4-mm1]# ls
> > Killed
> > //above are caused by OOM.
> > [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
> > 32763
> > [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
> > 32768
> > // fully filled by page cache. no reclaim run.
> > ==
> >
> > The reason this happens is because I used kernelcore= boot option, i.e
> > ZONE_MOVABLE. Seems try_to_free_mem_container_pages() ignores ZONE_MOVABLE.
> >
> > Quick fix is attached, but Mel's one-zonelist-pernode patch may change this.
> > I'll continue to watch.
> >
> > Thanks,
> > -Kame
> > ==
> > Now, there is ZONE_MOVABLE...
> >

> > page cache and user pages are allocated from gfp_zone(GFP_HIGHUSER_MOVABLE)
> >
> > Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> > ---
> > mm/vmscan.c | 9 ++-------
> > 1 file changed, 2 insertions(+), 7 deletions(-)
> >
> > Index: linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
> > ===
> > --- linux-2.6.23-rc4-mm1.bak.orig/mm/vmscan.c
> > +++ linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
> > @@ -1351,12 +1351,6 @@ unsigned long try_to_free_pages(struct z
> >
> > #ifdef CONFIG_CONTAINER_MEM_CONT
> >
> > -#ifdef CONFIG_HIGHMEM
> > -#define ZONE_USERPAGES ZONE_HIGHMEM
> > -#else
> > -#define ZONE_USERPAGES ZONE_NORMAL
> > -#endif
> > -
> > unsigned long try_to_free_mem_container_pages(struct mem_container *mem_cont)
> > {
> >   struct scan_control sc = {
> > @@ -1371,9 +1365,10 @@ unsigned long try_to_free_mem_container_
> >   };
> >   int node;
> >   struct zone **zones;
> > + int target_zone = gfp_zone(GFP_HIGHUSER_MOVABLE);
> >
> >   for_each_online_node(node) {
> > - zones = NODE_DATA(node)->node_zonelists[ZONE_USERPAGES].zones;
> > + zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
> >   if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
> >   return 1;
> >   }
>
> Mel, has sent out a fix (for the single zonelist) that conflicts with
> this one. Your fix looks correct to me, but it will be over ridden
> by Mel's fix (once those patches are in -mm).
>

"mel's fix" is rather too imprecise a term for me to make head or tail of this.

Oh well, the patch basically applied, so I whacked it in there, designated
as to be folded into memory-controller-make-charging-gfp-mask-aware.patch

_____
Containers mailing list

Subject: Re: problem with ZONE_MOVABLE.
Posted by Balbir Singh on Sat, 15 Sep 2007 06:14:04 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Thu, 13 Sep 2007 16:00:06 +0530
> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
>> KAMEZAWA Hiroyuki wrote:
>>> Hi,
>>>
>>> While I'm playing with memory controller of 2.6.23-rc4-mm1, I met following.
>>>
>>> ==
>>> [root@drpq test-2.6.23-rc4-mm1]# echo $$ > /opt/mem_control/group_1/tasks
>>> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
>>> 32768
>>> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
>>> 286
>>> // Memory is limited to 512 GiB. try "dd" 1GiB (page size is 16KB)
>>>
>>> [root@drpq test-2.6.23-rc4-mm1]# dd if=/dev/zero of=/tmp/tmpfile bs=1024 count=1048576
>>> Killed
>>> [root@drpq test-2.6.23-rc4-mm1]# ls
>>> Killed
>>> //above are caused by OOM.
>>> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.usage
>>> 32763
>>> [root@drpq test-2.6.23-rc4-mm1]# cat /opt/mem_control/group_1/memory.limit
>>> 32768
>>> // fully filled by page cache. no reclaim run.
>>> ==
>>>
>>> The reason this happens is because I used kernelcore= boot option, i.e
>>> ZONE_MOVABLE. Seems try_to_free_mem_container_pages() ignores ZONE_MOVABLE.
>>>
>>> Quick fix is attached, but Mel's one-zonelist-pernode patch may change this.
>>> I'll continue to watch.
>>>
>>> Thanks,
>>> -Kame
>>> ==
>>> Now, there is ZONE_MOVABLE...
>>>

>>> page cache and user pages are allocated from gfp_zone(GFP_HIGHUSER_MOVABLE)
>>>
>>> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
>>> ---
>>> mm/vmscan.c | 9 ++-------
>>> 1 file changed, 2 insertions(+), 7 deletions(-)
>>>
>>> Index: linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
>>> ===
>>> --- linux-2.6.23-rc4-mm1.bak.orig/mm/vmscan.c
>>> +++ linux-2.6.23-rc4-mm1.bak/mm/vmscan.c
>>> @@ -1351,12 +1351,6 @@ unsigned long try_to_free_pages(struct z
>>>
>>> #ifdef CONFIG_CONTAINER_MEM_CONT
>>>
>>> -#ifdef CONFIG_HIGHMEM
>>> -#define ZONE_USERPAGES ZONE_HIGHMEM
>>> -#else
>>> -#define ZONE_USERPAGES ZONE_NORMAL
>>> -#endif
>>> -
>>> unsigned long try_to_free_mem_container_pages(struct mem_container *mem_cont)
>>> {
>>>   struct scan_control sc = {
>>> @@ -1371,9 +1365,10 @@ unsigned long try_to_free_mem_container_
>>>   };
>>>   int node;
>>>   struct zone **zones;
>>> + int target_zone = gfp_zone(GFP_HIGHUSER_MOVABLE);
>>>
>>>   for_each_online_node(node) {
>>> - zones = NODE_DATA(node)->node_zonelists[ZONE_USERPAGES].zones;
>>> + zones = NODE_DATA(node)->node_zonelists[target_zone].zones;
>>>   if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
>>>   return 1;
>>>   }
>> Mel, has sent out a fix (for the single zonelist) that conflicts with
>> this one. Your fix looks correct to me, but it will be over ridden
>> by Mel's fix (once those patches are in -mm).
>>
>
> "mel's fix" is rather too imprecise a term for me to make head or tail of this.
>
> Oh well, the patch basically applied, so I whacked it in there, designated
> as to be folded into memory-controller-make-charging-gfp-mask-aware.patch

I agree that this fix is required and may be over-written by Mel'ls
patches in the future, but for now this is the correct fix. Thanks

for applying 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