Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Dave Hansen on Wed, 16 Aug 2006 19:59:29 GMT
View Forum Message <> Reply to Message

On Wed, 2006-08-16 at 12:15 -0700, Rohit Seth wrote:
> My preference would be to have container (I keep on saying container,
> but resource beancounter) pointer embeded in task, mm(not sure),
> address_space and anon_vma structures.

Hmm. If we can embed it in the mm, then we can get there from any given anon_vma (or any pte for that matter). Here's a little prototype for doing just that:

http://www.sr71.net/patches/2.6.18/2.6.18-rc4-mm1-lxc1/broke n-out/modify-lru-walk.patch

See file/anon_page_has_naughty_cpuset(). Anybody see any basic problems with doing it that way?

One trick with putting it in an mm is that we don't have a direct relationship between processes and mm's. We could also potentially have two different threads of a process in two different accounting contexts. But, that might be as simple to fix as disallowing things that share mms from being in different accounting contexts, unless you unshare the mm.

-- Dave

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Alan Cox on Thu, 17 Aug 2006 00:04:07 GMT
View Forum Message <> Reply to Message

Ar Mer, 2006-08-16 am 12:59 -0700, ysgrifennodd Dave Hansen:
> relationship between processes and mm's. We could also potentially have
> two different threads of a process in two different accounting contexts.
> But, that might be as simple to fix as disallowing things that share mms
> from being in different accounting contexts, unless you unshare the mm.

At the point I have twenty containers containing 20 copies of glibc to meet your suggestion it would be *far* cheaper to put it in the page struct.

Alan

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Dave Hansen on Thu, 17 Aug 2006 14:26:28 GMT
View Forum Message <> Reply to Message

On Thu, 2006-08-17 at 01:24 +0100, Alan Cox wrote:
> Ar Mer, 2006-08-16 am 12:59 -0700, ysgrifennodd Dave Hansen:
> > relationship between processes and mm's. We could also potentially have
> > two different threads of a process in two different accounting contexts.
> > But, that might be as simple to fix as disallowing things that share mms
> > from being in different accounting contexts, unless you unshare the mm.
>
> At the point I have twenty containers containing 20 copies of glibc to
> meet your suggestion it would be *far* cheaper to put it in the page
> struct.

My main thought is that _everybody_ is going to have to live with the
entry in the 'struct page'. Distros ship one kernel for everybody, and
the cost will be paid by those not even using any kind of resource
control or containers.

That said, it sure is simpler to implement, so I'm all for it!

-- Dave

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Rik van Riel on Thu, 17 Aug 2006 15:19:11 GMT
View Forum Message <> Reply to Message

Dave Hansen wrote:

> My main thought is that _everybody_ is going to have to live with the
> entry in the 'struct page'. Distros ship one kernel for everybody, and
> the cost will be paid by those not even using any kind of resource
> control or containers.

Every userspace or page cache page will be in an object
though. Could we do the pointer on a per object (mapping,
anon vma, ...) basis?

Kernel pages are not using all of their struct page entries,
so we could overload a field.

It all depends on how much we really care about not growing
struct page :)

--
What is important? What you want to be true, or what is true?

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)

Posted by Rohit Seth on Thu, 17 Aug 2006 16:31:48 GMT
View Forum Message <> Reply to Message

On Wed, 2006-08-16 at 12:59 -0700, Dave Hansen wrote:
> On Wed, 2006-08-16 at 12:15 -0700, Rohit Seth wrote:
> > My preference would be to have container (I keep on saying container,
> > but resource beancounter) pointer embeded in task, mm(not sure),
> > address_space and anon_vma structures.
>
> Hmm. If we can embed it in the mm, then we can get there from any given
> anon_vma (or any pte for that matter). Here's a little prototype for
> doing just that:
>
> http://www.sr71.net/patches/2.6.18/2.6.18-rc4-mm1-lxc1/broke n-out/modify-lru-walk.patch
>
> See file/anon_page_has_naughty_cpuset(). Anybody see any basic problems
> with doing it that way?
>
> One trick with putting it in an mm is that we don't have a direct
> relationship between processes and mm's. We could also potentially have
> two different threads of a process in two different accounting contexts.
> But, that might be as simple to fix as disallowing things that share mms
> from being in different accounting contexts, unless you unshare the mm.

But anon_vmas could be shared across different processes (with different
mms).

-rohit

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Rohit Seth on Thu, 17 Aug 2006 16:42:42 GMT
View Forum Message <> Reply to Message

On Thu, 2006-08-17 at 01:24 +0100, Alan Cox wrote:
> Ar Mer, 2006-08-16 am 12:59 -0700, ysgrifennodd Dave Hansen:
> > > relationship between processes and mm's. We could also potentially have
> > > two different threads of a process in two different accounting contexts.
> > > But, that might be as simple to fix as disallowing things that share mms
> > > from being in different accounting contexts, unless you unshare the mm.
>
> At the point I have twenty containers containing 20 copies of glibc to
> meet your suggestion it would be *far* cheaper to put it in the page
> struct.

I think the best would be to have a container for /usr/lib or /lib in

this case where you can account all pages belonging to files present in
these directories and shared across multiple applications.

-rohit

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Rohit Seth on Thu, 17 Aug 2006 17:16:43 GMT
View Forum Message <> Reply to Message

On Thu, 2006-08-17 at 07:26 -0700, Dave Hansen wrote:
> On Thu, 2006-08-17 at 01:24 +0100, Alan Cox wrote:
> > Ar Mer, 2006-08-16 am 12:59 -0700, ysgrifennodd Dave Hansen:
> > > relationship between processes and mm's. We could also potentially have
> > > two different threads of a process in two different accounting contexts.
> > > But, that might be as simple to fix as disallowing things that share mms
> > > from being in different accounting contexts, unless you unshare the mm.
> >
> > At the point I have twenty containers containing 20 copies of glibc to
> > meet your suggestion it would be *far* cheaper to put it in the page
> > struct.
>
> My main thought is that _everybody_ is going to have to live with the
> entry in the 'struct page'. Distros ship one kernel for everybody, and
> the cost will be paid by those not even using any kind of resource
> control or containers.
>
> That said, it sure is simpler to implement, so I'm all for it!

hmm, not sure why it is simpler.

-rohit

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Dave Hansen on Thu, 17 Aug 2006 17:23:21 GMT
View Forum Message <> Reply to Message

On Thu, 2006-08-17 at 10:16 -0700, Rohit Seth wrote:
> > That said, it sure is simpler to implement, so I'm all for it!
>
> hmm, not sure why it is simpler.

When you ask the question, "which container owns this page?", you don't
have to look far, nor is it ambiguous in any way. It is very strict,
and very straightforward.

-- Dave

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Rohit Seth on Thu, 17 Aug 2006 17:28:20 GMT
View Forum Message <> Reply to Message

On Thu, 2006-08-17 at 11:19 -0400, Rik van Riel wrote:
> Dave Hansen wrote:
>
> > My main thought is that _everybody_ is going to have to live with the
> > entry in the 'struct page'. Distros ship one kernel for everybody, and
> > the cost will be paid by those not even using any kind of resource
> > control or containers.
>
> Every userspace or page cache page will be in an object
> though. Could we do the pointer on a per object (mapping,
> anon vma, ...) basis?
>
> Kernel pages are not using all of their struct page entries,
> so we could overload a field.
>

Bingo. Even though it has the word "overload".

> It all depends on how much we really care about not growing
> struct page :)
>

Besides, if we have the container pointer based on address_space (for example) then it will also allow file based tracking...

I think page based container pointer makes more sense when you have container as the central part of page lists (in place of nodes) deciding which list the free page is going to come from, and when freed which list it is going to go back to.

-rohit

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Andi Kleen on Thu, 17 Aug 2006 17:35:36 GMT
View Forum Message <> Reply to Message

On Thursday 17 August 2006 19:28, Rohit Seth wrote:
> On Thu, 2006-08-17 at 11:19 -0400, Rik van Riel wrote:

> > Dave Hansen wrote:
> > > My main thought is that _everybody_ is going to have to live with the
> > > entry in the 'struct page'. Distros ship one kernel for everybody, and
> > > the cost will be paid by those not even using any kind of resource
> > > control or containers.
> >
> > Every userspace or page cache page will be in an object
> > though. Could we do the pointer on a per object (mapping,
> > anon vma, ...) basis?
> >
> > Kernel pages are not using all of their struct page entries,
> > so we could overload a field.
>
> Bingo. Even though it has the word "overload".

You would need to be careful. Both the rewritten slab and the new
tree network allocator use struct page fields already. There might
be conflicts already.

-Andi (who still doesn't see what's so bad about a separate table)

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Dave Hansen on Thu, 17 Aug 2006 17:49:23 GMT
View Forum Message <> Reply to Message

On Thu, 2006-08-17 at 20:43 +0200, Andi Kleen wrote:
> -Andi (who still doesn't see what's so bad about a separate table)

Not much wrong with that, as long as we do something more like sparsemem
than mem_map[]. ;)

-- Dave

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by dev on Fri, 18 Aug 2006 08:27:46 GMT
View Forum Message <> Reply to Message

Rik van Riel wrote:
> Dave Hansen wrote:
>
>> My main thought is that _everybody_ is going to have to live with the
>> entry in the 'struct page'. Distros ship one kernel for everybody, and
>> the cost will be paid by those not even using any kind of resource
>> control or containers.
>

>
> Every userspace or page cache page will be in an object
> though. Could we do the pointer on a per object (mapping,
> anon vma, ...) basis?
in this case no memory fractions accounting is possible :/
please, note, this field added by this patchset is in union
and used by user pages accounting as well.

> Kernel pages are not using all of their struct page entries,
> so we could overload a field.
yeah, we can. probably mapping.
but as I said we use the same pointer for user pages accounting as well.

> It all depends on how much we really care about not growing
> struct page :)
so what is your opinion?
Kernel compiled w/o UBC do not introduce additional pointer.

Thanks,
Kirill

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by dev on Fri, 18 Aug 2006 08:49:42 GMT
View Forum Message <> Reply to Message

Rohit Seth wrote:
> On Thu, 2006-08-17 at 07:26 -0700, Dave Hansen wrote:
>
>>On Thu, 2006-08-17 at 01:24 +0100, Alan Cox wrote:
>>
>>>Ar Mer, 2006-08-16 am 12:59 -0700, ysgrifennodd Dave Hansen:
>>>
>>>>relationship between processes and mm's. We could also potentially have
>>>>two different threads of a process in two different accounting contexts.
>>>>But, that might be as simple to fix as disallowing things that share mms
>>>>from being in different accounting contexts, unless you unshare the mm.
>>>
>>>At the point I have twenty containers containing 20 copies of glibc to
>>>meet your suggestion it would be *far* cheaper to put it in the page
>>>struct.
>>
>>My main thought is that _everybody_ is going to have to live with the
>>entry in the 'struct page'. Distros ship one kernel for everybody, and
>>the cost will be paid by those not even using any kind of resource
>>control or containers.
>>
>>That said, it sure is simpler to implement, so I'm all for it!

>
>
>
> hmm, not sure why it is simpler.
because introducing additonal lookups/hashes etc. is harder and
adds another source for possible mistakes.
we can always optimize it out if people insist.

Kirill

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by dev on Fri, 18 Aug 2006 08:49:57 GMT
View Forum Message <> Reply to Message

Rohit Seth wrote:
> On Thu, 2006-08-17 at 07:26 -0700, Dave Hansen wrote:
>
>>On Thu, 2006-08-17 at 01:24 +0100, Alan Cox wrote:
>>
>>>Ar Mer, 2006-08-16 am 12:59 -0700, ysgrifennodd Dave Hansen:
>>>
>>>>relationship between processes and mm's. We could also potentially have
>>>>two different threads of a process in two different accounting contexts.
>>>>But, that might be as simple to fix as disallowing things that share mms
>>>>from being in different accounting contexts, unless you unshare the mm.
>>>
>>>At the point I have twenty containers containing 20 copies of glibc to
>>>meet your suggestion it would be *far* cheaper to put it in the page
>>>struct.
>>
>>My main thought is that _everybody_ is going to have to live with the
>>entry in the 'struct page'. Distros ship one kernel for everybody, and
>>the cost will be paid by those not even using any kind of resource
>>control or containers.
>>
>>That said, it sure is simpler to implement, so I'm all for it!
>
>
>
> hmm, not sure why it is simpler.
because introducing additonal lookups/hashes etc. is harder and
adds another source for possible mistakes.
we can always optimize it out if people insist (by cost of slower accounting).

Kirill

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Alan Cox on Fri, 18 Aug 2006 14:39:01 GMT
View Forum Message <> Reply to Message

Ar Gwe, 2006-08-18 am 12:52 +0400, ysgrifennodd Kirill Korotaev:
> > hmm, not sure why it is simpler.
> because introducing additonal lookups/hashes etc. is harder and
> adds another source for possible mistakes.
> we can always optimize it out if people insist (by cost of slower accounting).

It ought to be cheap. Given each set of page structs is an array its a
simple subtract and divide (or with care and people try to pack them
nicely for cache lines - shift) to get to the parallel accounting array.

Alan

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Rohit Seth on Fri, 18 Aug 2006 17:06:16 GMT
View Forum Message <> Reply to Message

On Fri, 2006-08-18 at 12:29 +0400, Kirill Korotaev wrote:
> Rik van Riel wrote:
> > Dave Hansen wrote:
> >
>> My main thought is that _everybody_ is going to have to live with the
>> entry in the 'struct page'. Distros ship one kernel for everybody, and
>> the cost will be paid by those not even using any kind of resource
>> control or containers.
> >
> >
> > Every userspace or page cache page will be in an object
> > though. Could we do the pointer on a per object (mapping,
> > anon vma, ...) basis?
> in this case no memory fractions accounting is possible :/
> please, note, this field added by this patchset is in union
> and used by user pages accounting as well.
>
> > Kernel pages are not using all of their struct page entries,
> > so we could overload a field.
> yeah, we can. probably mapping.
> but as I said we use the same pointer for user pages accounting as well.
>
> > It all depends on how much we really care about not growing
> > struct page :)
> so what is your opinion?
> Kernel compiled w/o UBC do not introduce additional pointer.

As Andi pointed out earlier that slab and network codes are going to use
the mapping field (and you pointed out that some of this is allocated
out of context), so seems like for kernel accounting we will need
another field in page structure.

-rohit

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Dave Hansen on Fri, 18 Aug 2006 19:32:03 GMT
View Forum Message <> Reply to Message

On Fri, 2006-08-18 at 15:59 +0100, Alan Cox wrote:
> Ar Gwe, 2006-08-18 am 12:52 +0400, ysgrifennodd Kirill Korotaev:
> > > hmm, not sure why it is simpler.
> > because introducing additonal lookups/hashes etc. is harder and
> > adds another source for possible mistakes.
> > we can always optimize it out if people insist (by cost of slower accounting).
>
> It ought to be cheap. Given each set of page structs is an array its a
> simple subtract and divide (or with care and people try to pack them
> nicely for cache lines - shift) to get to the parallel accounting array.

I wish page structs were just a simple array. ;)

It will just be a bit more code, but we'll need this for the two other
memory models: sparsemem and discontigmem. For discontig, we'll just
need pointers in the pg_data_ts and, for sparsemem, we'll likely need
another pointer in the 'struct mem_section'.

This will effectively double the memory we need for sparsemem (because
we only use one pointer per SECTION_SIZE bytes of memory) but, that
should be just fine.

Is there ever any need to go from the accounting structure *back* to the
page? I guess that might be the hard part with keeping parallel arrays,
if we even need it.

The reverse lookups might introduce a bit more pain with sparsemem and
discontig because, right now, we use bits in page->flags to help us go
find the containing node or the correct mem_section for the page.

-- Dave

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)

Posted by Alan Cox on Fri, 18 Aug 2006 20:32:36 GMT
View Forum Message <> Reply to Message

Ar Gwe, 2006-08-18 am 12:32 -0700, ysgrifennodd Dave Hansen:
> > It ought to be cheap. Given each set of page structs is an array its a
> > simple subtract and divide (or with care and people try to pack them
> > nicely for cache lines - shift) to get to the parallel accounting array.
>
> I wish page structs were just a simple array. ;)

Note I very carefully said "each set of"

> It will just be a bit more code, but we'll need this for the two other
> memory models: sparsemem and discontigmem. For discontig, we'll just
> need pointers in the pg_data_ts and, for sparsemem, we'll likely need
> another pointer in the 'struct mem_section'.

Actually I don't believe this is true in either case. Change the code
which allocates the page arrays to allocate (+ sizeof(void *) *
pages_in_array on the end of each array when using UBC. The rest then
seems to come out naturally.

Alan

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by dev on Mon, 21 Aug 2006 09:41:56 GMT
View Forum Message <> Reply to Message

Alan Cox wrote:
> Ar Gwe, 2006-08-18 am 12:32 -0700, ysgrifennodd Dave Hansen:
>
>>>It ought to be cheap. Given each set of page structs is an array its a
>>>simple subtract and divide (or with care and people try to pack them
>>>nicely for cache lines - shift) to get to the parallel accounting array.
>>
>>I wish page structs were just a simple array. ;)
>
>
> Note I very carefully said "each set of"
>
>
>>It will just be a bit more code, but we'll need this for the two other
>>memory models: sparsemem and discontigmem. For discontig, we'll just
>>need pointers in the pg_data_ts and, for sparsemem, we'll likely need
>>another pointer in the 'struct mem_section'.
>
>
>

> Actually I don't believe this is true in either case. Change the code
> which allocates the page arrays to allocate (+ sizeof(void *) *
> pages_in_array on the end of each array when using UBC. The rest then
> seems to come out naturally.
I only doubt what gain we will have in this situation.
boot-time selectable vs. CONFIG-selectable?

Kirill
