
Subject: Re: [PATCH 3/5] page_cgroup: make page tracking available for blkio
Posted by [Andrea Righi](#) on Wed, 23 Feb 2011 08:59:11 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Wed, Feb 23, 2011 at 01:49:10PM +0900, KAMEZAWA Hiroyuki wrote:

> On Wed, 23 Feb 2011 00:37:18 +0100

> Andrea Righi <arighi@develer.com> wrote:

>

>> On Tue, Feb 22, 2011 at 06:06:30PM -0500, Vivek Goyal wrote:

>>> On Wed, Feb 23, 2011 at 12:01:47AM +0100, Andrea Righi wrote:

>>>> On Tue, Feb 22, 2011 at 01:01:45PM -0700, Jonathan Corbet wrote:

>>>>> On Tue, 22 Feb 2011 18:12:54 +0100

>>>>> Andrea Righi <arighi@develer.com> wrote:

>>>>>

>>>>>> The page_cgroup infrastructure, currently available only for the memory
>>>>>> cgroup controller, can be used to store the owner of each page and
>>>>>> opportunely track the writeback IO. This information is encoded in
>>>>>> the upper 16-bits of the page_cgroup->flags.

>>>>>>

>>>>>> A owner can be identified using a generic ID number and the following
>>>>>> interfaces are provided to store a retrieve this information:

>>>>>>

>>>>>> unsigned long page_cgroup_get_owner(struct page *page);

>>>>>> int page_cgroup_set_owner(struct page *page, unsigned long id);

>>>>>> int page_cgroup_copy_owner(struct page *npage, struct page *opage);

>>>>>>

>>>>>> My immediate observation is that you're not really tracking the "owner"
>>>>>> here - you're tracking an opaque 16-bit token known only to the block
>>>>>> controller in a field which - if changed by anybody other than the block
>>>>>> controller - will lead to mayhem in the block controller. I think it
>>>>>> might be clearer - and safer - to say "blkcg" or some such instead of
>>>>>> "owner" here.

>>>>>>

>>>>>>

>>>>>> Basically the idea here was to be as generic as possible and make this
>>>>>> feature potentially available also to other subsystems, so that cgroup
>>>>>> subsystems may represent whatever they want with the 16-bit token.
>>>>>> However, no more than a single subsystem may be able to use this feature
>>>>>> at the same time.

>>>>>>

>>>>>> I'm tempted to say it might be better to just add a pointer to your
>>>>>> throtl_grp structure into struct page_cgroup. Or maybe replace the
>>>>>> mem_cgroup pointer with a single pointer to struct css_set. Both of
>>>>>> those ideas, though, probably just add unwanted extra overhead now to gain
>>>>>> generality which may or may not be wanted in the future.

>>>>>>

>>>>>> The pointer to css_set sounds good, but it would add additional space to
>>>>>> the page_cgroup struct. Now, page_cgroup is 40 bytes (in 64-bit arch)

> > > and all of them are allocated at boot time. Using unused bits in
> > > page_cgroup->flags is a choice with no overhead from this point of view.
> > >
> > > I think John suggested replacing mem_cgroup pointer with css_set so that
> > > size of the structure does not increase but it leads extra level of
> > > indirection.
> >
> > OK, got it sorry.
> >
> > So, IIUC we save css_set pointer and get a struct cgroup as following:
> >
> > struct cgroup *cgrp = css_set->subsys[subsys_id]->cgroup;
> >
> > Then, for example to get the mem_cgroup reference:
> >
> > struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
> >
> > It seems a lot of indirections, but I may have done something wrong or
> > there could be a simpler way to do it.
> >
>
>
> Then, page_cgroup should have reference count on css_set and make tons of
> atomic ops.
>
> BTW, bits of pc->flags are used for storing sectionID or nodeID.
> Please clarify your 16bit never breaks that information. And please keep
> more 4-5 flags for dirty_ratio support of memcg.

OK, I didn't see the recent work about section and node id encoded in the pc->flags, thanks. So, it'd be probably better to rebase the patch to the latest mmotm to check all this stuff.

>
> I wonder I can make pc->mem_cgroup to be pc->memid(16bit), then,
> ==
> static inline struct mem_cgroup *get_memcg_from_pc(struct page_cgroup *pc)
> {
> struct cgroup_subsys_state *css = css_lookup(&mem_cgroup_subsys, pc->memid);
> return container_of(css, struct mem_cgroup, css);
> }
> ==
> Overhead will be seen at updating file statistics and LRU management.
>
> But, hmm, can't you do that tracking without page_cgroup ?
> Because the number of dirty/writeback pages are far smaller than total pages,
> chasing I/O with dynamic structure is not very bad..
>

> preparing [pfn -> blkio] record table and move that information to struct bio
> in dynamic way is very difficult ?

This would be ok for dirty pages, but consider that we're also tracking anonymous pages. So, if we want to control the swap IO we actually need to save this information for a lot of pages and at the end I think we'll basically duplicate the page_cgroup code.

Thanks,
-Andrea

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