Subject: Re: [-mm PATCH 6/9] Memory controller add per container LRU and reclaim (v4)
Posted by Vaidyanathan Srinivas on Tue, 07 Aug 2007 18:30:11 GMT
View Forum Message <> Reply to Message

Vaidyanathan Srinivasan wrote:
>
> YAMAMOTO Takashi wrote:
>>> +unsigned long mem_container_isolate_pages(unsigned long nr_to_scan,
>>> +   struct list_head *dst,
>>> +   unsigned long *scanned, int order,
>>> +   int mode, struct zone *z,
>>> +   struct mem_container *mem_cont,
>>> +   int active)
>>> +{
>>> + unsigned long nr_taken = 0;
>>> + struct page *page;
>>> + unsigned long scan;
>>> + LIST_HEAD(mp_list);
>>> + struct list_head *src;
>>> + struct meta_page *mp;
>>> +
>>> + if (active)
>>> + src = &mem_cont->active_list;
>>> + else
>>> + src = &mem_cont->inactive_list;
>>> +
>>> + for (scan = 0; scan < nr_to_scan && !list_empty(src); scan++) {
>>> + mp = list_entry(src->prev, struct meta_page, lru);
>> what prevents another thread from freeing mp here?
>
> mem_cont->lru_lock protects the list and validity of mp. If we hold
> mem_cont->lru_lock for this entire loop, then we preserve the validity
> of mp. However that will be holding up container charge and uncharge.
>
> This entire routing is called with zone->lru_lock held by the caller.
> So within a zone, this routine is serialized.
>
> However page uncharge may race with isolate page. But will that lead
> to any corruption of the list? We may be holding the lock for too
> much time just to be on the safe side.
>
> Please allow us some time to verify whether this is indeed inadequate
> locking that will lead to corruption of the list.

I did few runs and checked for ref_cnt on meta_page and there seems to be a race between isolate pages and uncharge. We will probably have to increase the ref_cnt on meta_page while we are isolating it. I am

trying to see if we can solve the problem by manipulating the ref_cnt
on the meta_page.

--Vaidy

> Thanks for pointing out this situation.
> --Vaidy
>
>>> + spin_lock(&mem_cont->lru_lock);
>>> + if (mp)
>>> +   page = mp->page;
>>> + spin_unlock(&mem_cont->lru_lock);
>>> + if (!mp)
>>> +   continue;
>> YAMAMOTO Takashi
>> _____
>> Containers mailing list
>> Containers@lists.linux-foundation.org
>> https://lists.linux-foundation.org/mailman/listinfo/containers
>>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>

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