Subject: Re: [PATCH 3/3][RFC] Containers: Pagecache controller reclaim
Posted by Vaidyanathan Srinivas on Tue, 27 Mar 2007 12:25:06 GMT
View Forum Message <> Reply to Message

Aubrey Li wrote:
> On 3/27/07, Vaidyanathan Srinivasan <svaidy@linux.vnet.ibm.com> wrote:
>>
>> Aubrey Li wrote:
>>> On 3/27/07, Vaidyanathan Srinivasan <svaidy@linux.vnet.ibm.com> wrote:
>>>> Correct, shrink_page_list() is called from shrink_inactive_list() but
>>>> the above code is patched in shrink_active_list(). The
>>>> 'force_reclaim_mapped' label is from function shrink_active_list() and
>>>> not in shrink_page_list() as it may seem in the patch file.
>>>>
>>>> While removing pages from active_list, we want to select only
>>>> pagecache pages and leave the remaining in the active_list.
>>>> page_mapped() pages are _not_ of interest to pagecache controller
>>>> (they will be taken care by rss controller) and hence we put it back.
>>>> Also if the pagecache controller is below limit, no need to reclaim
>>>> so we put back all pages and come out.
>>> Oh, I just read the patch, not apply it to my local tree, I'm working
>>> on 2.6.19 now.
>>> So the question is, when vfs pagecache limit is hit, the current
>>> implementation just reclaim few pages, so it's quite possible the
>>> limit is hit again, and hence the reclaim code will be called again
>>> and again, that will impact application performance.
>> Yes, you are correct. So if we start reclaiming one page at a time,
>> then the cost of reclaim is very high and we would be calling the
>> reclaim code too often. Hence we have a 'buffer zone' or 'reclaim
>> threshold' or 'push back' around the limit. In the patch we have a 64
>> page (256KB) NR_PAGES_RECLAIM_THRESHOLD:
>>
>> int pagecache_acct_shrink_used(unsigned long nr_pages)
>> {
>> unsigned long ret = 0;
>> atomic_inc(&reclaim_count);
>> +
>> + /* Don't call reclaim for each page above limit */
>> + if (nr_pages > NR_PAGES_RECLAIM_THRESHOLD) {
>> + ret += shrink_container_memory(
>> + RECLAIM_PAGECACHE_MEMORY, nr_pages, NULL);
>> + }
>> +
>> return 0;
>> }
>>
>> Hence we do not call the reclaimer if the threshold is exceeded by
>> just 1 page... we wait for 64 pages or 256KB of pagecache memory to go

>> overlimit and then call the reclaimer which will reclaim all 64 pages
>> in one shot.
>>
>> This prevents the reclaim code from being called too often and it also
>> keeps the cost of reclaim low.
>>
>> In future patches we are planing to have a percentage based reclaim
>> threshold so that it would scale well with the container size.
>>
> Actually it's not a good idea IMHO. No matter how big the threshold
> is, it's not suitable. If it's too small, application performance will
> be impacted seriously after pagecache limit is hit. If it's too large,
> Limiting pagecache is useless.
>
> Why not reclaim pages as much as possible when the pagecache limit is hit?
>

Well, that seems to be a good suggestion. We will try it out by
asking the reclaimer to do as much as possible in minimum time/effort.
 However we have to figure out how hard we want to push the reclaimer.
 In fact we can push the shrink_active_list() and
shrink_inactive_list() routines to reclaim the _all_ container pages.
 We do have reclaim priority to play with. Let see if we can comeup
with some automatic method to reclaim 'good' number of pages each time.

--Vaidy
