Subject: Re: [RFC][PATCH][3/4] Add reclaim support
Posted by Balbir Singh on Mon, 19 Feb 2007 11:16:33 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Mon, 19 Feb 2007 16:20:53 +0530 Balbir Singh <balbir@in.ibm.com> wrote:
>
>>>> + * so, is the container over it's limit. Returns 1 if the container is above
>>>> + * its limit.
>>>> + */
>>>> +int memctlr_mm_overlimit(struct mm_struct *mm, void *sc_cont)
>>>> +{
>>>> + struct container *cont;
>>>> + struct memctlr *mem;
>>>> + long usage, limit;
>>>> + int ret = 1;
>>>> +
>>>> + if (!sc_cont)
>>>> + goto out;
>>>> +
>>>> + read_lock(&mm->container_lock);
>>>> + cont = mm->container;
>>>> +
>>>> + /*
>>>> + * Regular reclaim, let it proceed as usual
>>>> + */
>>>> + if (!sc_cont)
>>>> + goto out;
>>>> +
>>>> + ret = 0;
>>>> + if (cont != sc_cont)
>>>> + goto out;
>>>> +
>>>> + mem = memctlr_from_cont(cont);
>>>> + usage = atomic_long_read(&mem->counter.usage);
>>>> + limit = atomic_long_read(&mem->counter.limit);
>>>> + if (limit && (usage > limit))
>>>> + ret = 1;
>>>> +out:
>>>> + read_unlock(&mm->container_lock);
>>>> + return ret;
>>>> +}
>>> hm, I wonder how much additional lock traffic all this adds.
>>>
>> It's a read_lock() and most of the locks are read_locks
>> which allow for concurrent access, until the container
>> changes or goes away
>
>

> read_lock isn't free, and I suspect we're calling this function pretty
> often (every pagefault?) It'll be measurable on some workloads, on some
> hardware.
>
> It probably won't be terribly bad because each lock-taking is associated
> with a clear_page(). But still, if there's any possibility of lightening
> the locking up, now is the time to think about it.
>

Yes, good point. I'll revisit to see if barriers can replace the locking
or if the locking is required at all?

>>>> @@ -66,6 +67,9 @@ struct scan_control {
>>>> int swappiness;
>>>>
>>>> int all_unreclaimable;
>>>> +
>>>> + void *container; /* Used by containers for reclaiming */
>>>> + /* pages when the limit is exceeded */
>>>> };
>>> eww. Why void*?
>>>
>> I did not want to expose struct container in mm/vmscan.c.
>
> It's already there, via rmap.h
>

Yes, true

>> An additional
>> thought was that no matter what container goes in the field would be
>> useful for reclaim.
>
> Am having trouble parsing that sentence ;)
>
>

The thought was that irrespective of the infrastructure that goes in
having an entry for reclaim in scan_control would be useful. I guess
the name exposes what the type tries to hide :-)

--
Warm Regards,
Balbir Singh