Subject: Re: [RFC][PATCH][0/4] Memory controller (RSS Control)
Posted by Balbir Singh on Mon, 19 Feb 2007 10:00:39 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Mon, 19 Feb 2007 12:20:19 +0530 Balbir Singh <balbir@in.ibm.com> wrote:
>
>> This patch applies on top of Paul Menage's container patches (V7) posted at
>>
>> http://lkml.org/lkml/2007/2/12/88
>>
>> It implements a controller within the containers framework for limiting
>> memory usage (RSS usage).
>
> It's good to see someone building on someone else's work for once, rather
> than everyone going off in different directions. It makes one hope that we
> might actually achieve something at last.
>

Thanks! It's good to know we are headed in the right direction.

```
>
> The key part of this patchset is the reclaim algorithm:
>
>> @@ -636,6 +642,15 @@ static unsigned long isolate_lru_pages(u
>>
>>    list_del(&page->lru);
>>    target = src;
>> +  /*
>> +    * For containers, do not scan the page unless it
>> +    * belongs to the container we are reclaiming for
>> +    */
>> +  if (container && !page_in_container(page, zone, container)) {
>> +   scan--;
>> +   goto done;
>> +  }
>
```

> Alas, I fear this might have quite bad worst-case behaviour. One small
> container which is under constant memory pressure will churn the
> system-wide LRUs like mad, and will consume rather a lot of system time.
> So it's a point at which container A can deleteriously affect things which
> are running in other containers, which is exactly what we're supposed to
> not do.
>

Hmm.. I guess it's space vs time then :-) A CPU controller could
control how much time is spent reclaiming ;)

Coming back, I see the problem you mentioned and we have been thinking
of several possible solutions. In my introduction I pointed out

"Come up with cool page replacement algorithms for containers
(if possible without any changes to struct page)"

The solutions we have looked at are

1. Overload the LRU list_head in struct page to have a global
   LRU + a per container LRU

```
+------+          prev      +------+
| page +--------------------->| page +---------
|  0   |<---------------------+  1   |<--------
+------+          next      +------+

              Global LRU

          +------+
   +------- + prev |
   |       | next +-------------------------------------------+
   |       +------+                                           |
   V          ^                                              V
+------+       |  prev      +------+                    +------+
| page +       +------------ + page +---------.....       | page +
|  0   |--------------------->+  1   |<--------           |  n   |
+------+          next      +------+                    +------+

              Global LRU + Container LRU
```

Page 1 and n belong to the same container, to get to page 0, you need
two de-references

2. Modify struct page to point to a container and allow each container to
   have a per-container LRU along with the global LRU

For efficiency we need the container LRU and we don't want to split
the global LRU either.

We need to optimize the reclaim path, but I thought of that as a secondary
problem. Once we all agree that the controller looks simple, accounts well
and works. We can/should definitely optimize the reclaim path.

--
Warm Regards,
Balbir Singh