

---

Subject: Re: [RFC] mm-controller  
Posted by [Pavel Emelianov](#) on Fri, 22 Jun 2007 02:21:54 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Balbir Singh wrote:

[snip]

>> With the current dual list approach, something like that could be done  
>> by treating the container lists as pure FIFO (and ignore the reference  
>> bit and all that) and make container reclaim only unmap, not write out  
>> pages.

>>

>> Then global reclaim will do the work (if needed), and containers get  
>> churn, equating the page ownership.

>>

>

> I did implement the unmap only logic for shared pages in version 2  
> of my RSS controller

>

> <http://lkml.org/lkml/2007/2/19/10>

>

> It can be added back if required quite easily. Pavel what do you think  
> about it?

I think it's wrong. Look, when the container hits the limit and just unmaps the pages the following situation may occur: some \*other\* container will hit the global shortage and will have to wait till the other's pages are flushed to disk. This is not a true isolation. If we send the pages to the disk right when the container hits the limit we spend its time, its IO bandwidth, etc and allow for others to have the free set of pages without additional efforts.

[snip]

>>>> Because, if the data is shared between containers isolation is broken anyway  
>>>> and we might as well charge them equally [1].

>>>>

>>>> Move the full reclaim structures from struct zone to these structures.

>>>>

>>>>

>>>> struct reclaim;

>>>>

>>>> struct reclaim\_zone {

>>>> spinlock\_t lru\_lock;

>>>>

>>>> struct list\_head active;

>>>> struct list\_head inactive;

```
>>>>
>>>> unsigned long nr_active;
>>>> unsigned long nr_inactive;
>>>>
>>>> struct reclaim *reclaim;
>>>> };
>>>>
>>>> struct reclaim {
>>>> struct reclaim_zone zone_reclaim[MAX_NR_ZONES];
>>>>
>>>> spinlock_t containers_lock;
>>>> struct list_head containers;
>>>> unsigned long nr_containers;
>>>> };
>>>>
>>>>
>>>> struct address_space {
>>>> ...
>>>> struct reclaim reclaim;
>>>> };
>>>>
```

Peter, could you prepare some POC patches instead? See, when looking at the patches is simpler to understand what is going on then when reading the plain text. Moreover, when making the patches some unexpected details of the kernel internals arise and the ideas begin to change...

[snip]

---