Subject: Re: [PATCH 0/8] RSS controller based on process containers (v2)
Posted by Peter Zijlstra on Mon, 09 Apr 2007 15:54:50 GMT
View Forum Message <> Reply to Message

ugh /me no like.

The basic premises seems to be that we can track page owners perfectly
(although this patch set does not yet do so), through get/release
operations (on _mapcount).

This is simply not true for unmapped pagecache pages. Those receive no
'release' event; (the usage by find_get_page() could be seen as 'get').

Also, you don't seem to balance the active/inactive scanning on a per
container basis. This skews the per container working set logic.

Lastly, you don't call the slab shrinker for container reclaim; which
would leave slab reclaim only for those few non process specific
allocations, which would greatly skew the pagecache/slab balance.

Let us call

```
struct reclaim_struct {
 struct list_head active_list;
 struct list_head inactive_list;
 unsigned long nr_active;
 unsigned long nr_inactive;
}
```

Lets recognise three distinct page categories:
 - anonymous memory,
 - mapped pagecache, and
 - unmapped pagecache.

We then keep anonymous pages on a per container reclaim_struct, these
pages are fully accounted to each container.

We keep mapped pagecache pages on per inode reclaim_structs, these files
could be shared between containers and we could either just account all
pages belonging to each file proportional to the number of containers
involved, or do a more precise accounting.

We keep unmapped pagecache pages on a global reclaim_struct, these pages
can, in general, not be pinned to a specific container; all we can do is
keep a floating proportion relative to container 'get' events

(find_get_page() and perhaps add_to_page_cache()).

Reclaim will then have to fairly reclaim pages from all of these lists.
If we schedule such that it appears that these lists are parallel
instead of serial - that is a each tail is really a tail, not the head
of another list - the current reclaim semantics are preserved.

The slab shrinker should be called proportional to the containers size
relative to the machine.

Global reclaim will have to call each container reclaim in proportional
fashion.

The biggest problem with this approach is that there is no per zone
reclaim left, which is relied upon by the allocator to provide free
pages in a given physical address range. However there has been talk to
create a proper range allocator independent of zones.

Just my 0.02 euro..

Peter
