Subject: Re: [Fwd: [PATCH -RSS 1/1] Fix reclaim failure]
Posted by xemul on Tue, 05 Jun 2007 07:16:32 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> Sorry forgot to CC you.
>
> -------- Original Message --------
> Subject: [PATCH -RSS 1/1] Fix reclaim failure
> Date: Mon, 04 Jun 2007 21:02:44 +0530
> From: Balbir Singh <balbir@linux.vnet.ibm.com>
> To: Andrew Morton <akpm@osdl.org>
> CC: Linux Containers <containers@lists.osdl.org>,	Balbir Singh
<balbir@linux.vnet.ibm.com>,	Vaidyanathan Srinivasan <svaidy@linux.vnet.ibm.com>,
Linux Kernel Mailing List <linux-kernel@vger.kernel.org>
>
>
> This patch fixes the problem seen when a container goes over its limit, the
> reclaim is unsuccessful and the application is terminated. The problem
> is that all pages are by default added to the active list of the RSS
> controller. When __isolate_lru_page() is called, it checks to see if
> the list that the page is on (active or inactive) is the same as
> what PageActive(page) returns. If this is not the case, the page is skipped.
> In our case a page might not have the PG_active bit set and might be on
> the active list of the container, thus we were ignoring those pages and
> our reclaim fails, leading to the application being killed.
>
> Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
> ---
>
```
>  mm/rss_container.c |   31 +++++++++++++++++++++++--------
>  1 file changed, 23 insertions(+), 8 deletions(-)
>
> diff -puN mm/rss_container.c~rss-fix-free-of-active-pages mm/rss_container.c
> --- linux-2.6.22-rc2-mm1/mm/rss_container.c~rss-fix-free-of-active-pages 2007-06-04
19:48:56.000000000 +0530
> +++ linux-2.6.22-rc2-mm1-balbir/mm/rss_container.c 2007-06-04 19:50:18.000000000 +0530
> @@ -205,18 +205,37 @@ void container_rss_move_lists(struct pag
>  }
>
>  static unsigned long isolate_container_pages(unsigned long nr_to_scan,
> -  struct list_head *src, struct list_head *dst,
> -  unsigned long *scanned, struct zone *zone, int mode)
> +  struct rss_container *rss, struct list_head *dst,
> +  unsigned long *scanned, struct zone *zone, int mode,
> +  int active)
>  {
>   unsigned long nr_taken = 0;
>   struct page *page;
>   struct page_container *pc;
>   unsigned long scan;
>   LIST_HEAD(pc_list);
> + struct list_head *src;
> +
> +  src = active ? &rss->active_list : &rss->inactive_list;
>
>   for (scan = 0; scan < nr_to_scan && !list_empty(src); scan++) {
>    pc = list_entry(src->prev, struct page_container, list);
>    page = pc->page;
> +
> +  /*
> +   * We might have got our active, inactive lists
> +   * incorrect, fix it here
> +   */
> +  if (active && !PageActive(page)) {
> +   list_move(&pc->list, &rss->inactive_list);
> +   scan--;
> +   continue;
> +  } else if (!active && PageActive(page)) {
> +   list_move(&pc->list, &rss->active_list);
> +   scan--;
> +   continue;
> +  }
> +
```

Actually the plan was to keep these lists consistent, i.e. when page
drops the active bit and moves to the inactive global LRU list, the
according page_container should be migrated as well. Where's the place
that messes the lists? I thought I found all the places when the page
migrates across the lists...

```
>    /*
>     * TODO: now we hold all the pages in one... ok, two lists
>     * and skip the pages from another zones with the check
> @@ -249,12 +268,8 @@ unsigned long isolate_pages_in_container
>
>   /* we are called with zone->lru_lock held with irqs disabled */
>   spin_lock(&rss->res.lock);
> - if (active)
> -  ret = isolate_container_pages(nr_to_scan, &rss->active_list,
> -    dst, scanned, zone, mode);
> - else
> -  ret = isolate_container_pages(nr_to_scan, &rss->inactive_list,
> -    dst, scanned, zone, mode);
> + ret = isolate_container_pages(nr_to_scan, rss, dst, scanned, zone,
> +     mode, active);
```

I wanted to keep the solution of what list to select here to make it
easier to switch to per-zone containers lists. With this check moved
to the actual isolation function we won't be able to isolate pages from
arbitrary list if we need such, but I believe we will need.

```
>   spin_unlock(&rss->res.lock);
>   return ret;
>  }
> _
>
```

Thanks,
Pavel

Subject: Re: [Fwd: [PATCH -RSS 1/1] Fix reclaim failure]
Posted by Balbir Singh on Tue, 05 Jun 2007 08:12:21 GMT

View Forum Message <> Reply to Message

Pavel Emelianov wrote:

```
>>
>>  static unsigned long isolate_container_pages(unsigned long nr_to_scan,
>> -  struct list_head *src, struct list_head *dst,
>> -  unsigned long *scanned, struct zone *zone, int mode)
>> +  struct rss_container *rss, struct list_head *dst,
>> +  unsigned long *scanned, struct zone *zone, int mode,
>> +  int active)
>>  {
>>   unsigned long nr_taken = 0;
>>   struct page *page;
>>   struct page_container *pc;
>>   unsigned long scan;
>>   LIST_HEAD(pc_list);
>> + struct list_head *src;
>> +
>> +  src = active ? &rss->active_list : &rss->inactive_list;
>>
>>   for (scan = 0; scan < nr_to_scan && !list_empty(src); scan++) {
>>    pc = list_entry(src->prev, struct page_container, list);
>>    page = pc->page;
>> +
>> +  /*
>> +   * We might have got our active, inactive lists
>> +   * incorrect, fix it here
>> +   */
>> +  if (active && !PageActive(page)) {
>> +   list_move(&pc->list, &rss->inactive_list);
>> +   scan--;
>> +   continue;
>> +  } else if (!active && PageActive(page)) {
>> +   list_move(&pc->list, &rss->active_list);
>> +   scan--;
>> +   continue;
>> +  }
>> +
>
```
> Actually the plan was to keep these lists consistent, i.e. when page
> drops the active bit and moves to the inactive global LRU list, the
> according page_container should be migrated as well. Where's the place
> that messes the lists? I thought I found all the places when the page
> migrates across the lists...
>

Yes, we do that. This fix is required for the situation occurs when a
page is brought in initially. A file backed page does not have it's
PG_active bit. Alternatively, we could modify the call sites to put
the page in the correct list (active/inactive), but that can easily
lead to complexity in the case the page is already on the LRU.

```
>>    /*
>>     * TODO: now we hold all the pages in one... ok, two lists
>>     * and skip the pages from another zones with the check
>> @@ -249,12 +268,8 @@ unsigned long isolate_pages_in_container
>>
>>   /* we are called with zone->lru_lock held with irqs disabled */
>>   spin_lock(&rss->res.lock);
>> - if (active)
>> -  ret = isolate_container_pages(nr_to_scan, &rss->active_list,
>> -    dst, scanned, zone, mode);
>> - else
>> -  ret = isolate_container_pages(nr_to_scan, &rss->inactive_list,
>> -    dst, scanned, zone, mode);
>> + ret = isolate_container_pages(nr_to_scan, rss, dst, scanned, zone,
>> +     mode, active);
>
```
> I wanted to keep the solution of what list to select here to make it
> easier to switch to per-zone containers lists. With this check moved
> to the actual isolation function we won't be able to isolate pages from
> arbitrary list if we need such, but I believe we will need.
>

Hmm.. if we change adding back the pages correctly in the call site, this
change can be avoided.

```
>>   spin_unlock(&rss->res.lock);
>>   return ret;
>>  }
>> _
>>
>
```

> Thanks,
> Pavel

--
 Warm Regards,
 Balbir Singh
 Linux Technology Center
 IBM, ISTL