
Subject: [RFC][PATCH 1/3] Pagecache accounting
Posted by [Vaidyanathan Srinivas](#) on Fri, 29 Jun 2007 06:19:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Pagecache Accounting

The rss accounting hooks have been generalised to handle both anon pages and file backed pages and charge the resource counter.

Ref count has been added to page_container structure. The ref count is used to ensure a page is added or removed from page_container list only once independent of repeated calls from pagecache, swapcache and mmap to RSS.

No setup patch is required since rss_limit and rss_usage has been generalised as the resource counter for pagecache as well.

Signed-off-by: Vaidyanathan Srinivasan <svaidy@linux.vnet.ibm.com>

```
---  
include/linux/rss_container.h | 18 +----  
mm/rss_container.c          | 134 ++++++-----  
2 files changed, 99 insertions(+), 53 deletions(-)
```

```
--- linux-2.6.22-rc2-mm1.orig/include/linux/rss_container.h  
+++ linux-2.6.22-rc2-mm1/include/linux/rss_container.h  
@@ -68,11 +68,11 @@ struct rss_container;  
 *  
 */
```

```
-int container_rss_prepare(struct page *, struct vm_area_struct *vma,  
+int container_page_prepare(struct page *, struct mm_struct *mm,  
    struct page_container **);  
-void container_rss_add(struct page_container *);  
-void container_rss_del(struct page_container *);  
-void container_rss_release(struct page_container *);  
+void container_page_add(struct page_container *);  
+void container_page_del(struct page_container *);  
+void container_page_release(struct page_container *);  
  
void container_out_of_memory(struct rss_container *);
```

```
@@ -85,22 +85,22 @@ unsigned long isolate_pages_in_container  
    int order, int mode, struct zone *zone,  
    struct rss_container *, int active);  
#else  
-static inline int container_rss_prepare(struct page *pg,  
-    struct vm_area_struct *vma, struct page_container **pc)  
+static inline int container_page_prepare(struct page *pg,
```

```

+ struct mm_struct *mm, struct page_container **pc)
{
 *pc = NULL; /* to make gcc happy */
 return 0;
}

-static inline void container_rss_add(struct page_container *pc)
+static inline void container_page_add(struct page_container *pc)
{
}

-static inline void container_rss_del(struct page_container *pc)
+static inline void container_page_del(struct page_container *pc)
{
}

-static inline void container_rss_release(struct page_container *pc)
+static inline void container_page_release(struct page_container *pc)
{
}

--- linux-2.6.22-rc2-mm1.orig/mm/rss_container.c
+++ linux-2.6.22-rc2-mm1/mm/rss_container.c
@@ -56,6 +56,9 @@ struct rss_container {
 */

struct page_container {
+ unsigned long ref_cnt; /* Ref cnt to keep track of
+   * multiple additions of same page
+   */
    struct page *page;
    struct rss_container *cnt;
    struct list_head list; /* this is the element of (int)active_list of
@@ -93,26 +96,36 @@ void mm_free_container(struct mm_struct
 * I bet you have already read the comment in include/linux/rss_container.h :)
 */

-int container_rss_prepare(struct page *page, struct vm_area_struct *vma,
+int container_page_prepare(struct page *page, struct mm_struct *mm,
    struct page_container **ppc)
{
- struct rss_container *rss;
- struct page_container *pc;
-
- rCU_read_lock();
- rss = rCU_dereference(vma->vm_mm->rss_container);
- css_get(&rss->css);
- rCU_read_unlock();
}

```

```

-
- pc = kmalloc(sizeof(struct page_container), GFP_KERNEL);
- if (pc == NULL)
- goto out_nomem;
+ struct rss_container *rss;
+ struct page_container *pc;
+ int rc = 1;
+
+ /* Page may have been added to container earlier */
+ pc = page_container(page);
+ /* Check if this is fist time addition or not */
+ if (!pc) {
+ rCU_read_lock();
+ rss = rCU_dereference(mm->rss_container);
+ css_get(&rss->css);
+ rCU_read_unlock();
+ } else {
+ rss = pc->cnt;
+ }
-
- while (res_counter_charge(&rss->res, 1)) {
- if (try_to_free_pages_in_container(rss)) {
- atomic_inc(&rss->rss_reclaimed);
- continue;
- }
+ /* Charge the respective resource count first time only */
+ while (rc && !pc) {
+ rc = res_counter_charge(&rss->res, 1);
+
+ if (!rc)
+ break; /* All well */
+
+ if (try_to_free_pages_in_container(rss)) {
+ atomic_inc(&rss->rss_reclaimed);
+ continue; /* Try agin to charge container */
+ }
+
/* 
 * try_to_free_pages() might not give us a full picture
@@ -125,60 +138,93 @@ int container_rss_prepare(struct page *p
 if (res_counter_check_under_limit(&rss->res))
 continue;
-
- container_out_of_memory(rss);
- if (test_thread_flag(TIF_MEMDIE))
- goto out_charge;
- }
+ /* Unable to free memory?? */

```

```

+ container_out_of_memory(rss);
+ if (test_thread_flag(TIF_MEMDIE))
+ goto out_charge;
+ }
+
+ /* First time addition to container: Create new page_container */
+ if (!pc) {
+ pc = kzalloc(sizeof(struct page_container), GFP_KERNEL);
+ if (pc == NULL)
+ goto out_nomem;
+
+ pc->page = page;
+ pc->cnt = rss;
+ }

- pc->page = page;
- pc->cnt = rss;
- *ppc = pc;
- return 0;
+ *ppc = pc;
+ return 0;

out_charge:
- kfree(pc);
+ /* Need to zero page_container?? */
out_nomem:
- css_put(&rss->css);
- return -ENOMEM;
+ css_put(&rss->css);
+ return -ENOMEM;
+
}

void container_rss_release(struct page_container *pc)
+void container_page_release(struct page_container *pc)
{
    struct rss_container *rss;

    rss = pc->cnt;
- res_counter_uncharge(&rss->res, 1);
- css_put(&rss->css);
- kfree(pc);
+ /* Setting the accounts right */
+ if (!pc->ref_cnt) {
+ res_counter_uncharge(&rss->res, 1);
+ set_page_container(pc->page, NULL);
+ kfree(pc);
+ css_put(&rss->css);

```

```

+ }
}

-void container_rss_add(struct page_container *pc)
+void container_page_add(struct page_container *pc)
{
    struct page *pg;
    struct rss_container *rss;
+ unsigned long irqflags;

    pg = pc->page;
- rss = pc->cnt;
+ if (pg == ZERO_PAGE(0))
+     return;

- spin_lock_irq(&rss->res.lock);
- list_add(&pc->list, &rss->active_list);
- spin_unlock_irq(&rss->res.lock);
+ rss = pc->cnt;
+ spin_lock_irqsave(&rss->res.lock, irqflags);
+ if (!pc->ref_cnt)
+     list_add(&pc->list, &rss->inactive_list);
+ pc->ref_cnt++;
+ spin_unlock_irqrestore(&rss->res.lock, irqflags);

    set_page_container(pg, pc);
}

-void container_rss_del(struct page_container *pc)
+void container_page_del(struct page_container *pc)
{
+ struct page *page;
    struct rss_container *rss;
+ unsigned long irqflags;

+ page = pc->page;
    rss = pc->cnt;
- spin_lock_irq(&rss->res.lock);
- list_del(&pc->list);
- res_counter_uncharge_locked(&rss->res, 1);
- spin_unlock_irq(&rss->res.lock);

- css_put(&rss->css);
- kfree(pc);
+ if (page == ZERO_PAGE(0))
+     return;
+
+ spin_lock_irqsave(&rss->res.lock, irqflags);

```

```
+ pc->ref_cnt--;
+ if (!pc->ref_cnt) {
+   list_del(&pc->list);
+   set_page_container(page, NULL);
+ }
+ spin_unlock_irqrestore(&rss->res.lock, irqflags);
+
+ if (!pc->ref_cnt) {
+   res_counter_uncharge(&rss->res, 1);
+   kfree(pc);
+   css_put(&rss->css);
+ }
+
}

/*

```
