Subject: [RFC][PATCH 2/3] Containers: Pagecache accounting and control subsystem (v3)
Posted by Vaidyanathan Srinivas on Wed, 23 May 2007 14:52:17 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 respective resource counters.

New flags and 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.

Flags in page_container is used to keep the accounting correct between RSS and
unmapped pagecache (includes swapcache) pages.

Signed-off-by: Vaidyanathan Srinivasan <svaidy@linux.vnet.ibm.com>
---
 include/linux/rss_container.h | 19 +++-
 mm/rss_container.c | 161 ++++++++++++++++++++++++++++++++----------
 2 files changed, 127 insertions(+), 53 deletions(-)

--- linux-2.6.20.orig/include/linux/rss_container.h
+++ linux-2.6.20/include/linux/rss_container.h
@@ -12,13 +12,22 @@
 struct page_container;
 struct rss_container;

+/* Flags for container_page_xxx calls */
+#define PAGE_TYPE_RSS 0x00000001
+#define PAGE_TYPE_PAGECACHE 0x00000002
+#define PAGE_TYPE_SWAPCACHE 0x00000004
+
+#define PAGE_SUBTYPE_ANON 0x00010000
+#define PAGE_SUBTYPE_FILE 0x00020000
+
 #ifdef CONFIG_RSS_CONTAINER
-int container_rss_prepare(struct page *, struct vm_area_struct *vma,
- struct page_container **);
+int container_page_prepare(struct page *page, struct mm_struct *mm,
+ struct page_container **ppc, unsigned int flags);
+
+void container_page_add(struct page_container *, unsigned int flags);
+void container_page_del(struct page_container *, unsigned int flags);
+void container_page_release(struct page_container *, unsigned int flags);

```
-void container_rss_add(struct page_container *);
-void container_rss_del(struct page_container *);
-void container_rss_release(struct page_container *);
 void container_out_of_memory(struct rss_container *);

 void mm_init_container(struct mm_struct *mm, struct task_struct *tsk);
--- linux-2.6.20.orig/mm/rss_container.c
+++ linux-2.6.20/mm/rss_container.c
@@ -24,11 +24,18 @@ struct rss_container {
 };

 struct page_container {
+ unsigned long flags;
+ unsigned long ref_cnt;
  struct page *page;
  struct rss_container *cnt;
  struct list_head list;
 };

+/* Flags for Page Container */
+#define PAGE_IN_PAGECACHE  0x0001
+#define PAGE_IN_RSS   0x0002
+#define PAGE_IN_SWAPCACHE  0x0004
+
 static inline struct rss_container *rss_from_cont(struct container *cnt)
 {
  return container_of(container_subsys_state(cnt, rss_subsys_id),
@@ -49,52 +56,143 @@ void mm_free_container(struct mm_struct
  css_put(&mm->rss_container->css);
 }

-int container_rss_prepare(struct page *page, struct vm_area_struct *vma,
-  struct page_container **ppc)
+int container_page_prepare(struct page *page, struct mm_struct *mm,
+  struct page_container **ppc, unsigned int flags)
 {
  struct rss_container *rss;
  struct page_container *pc;
+ int rc = 1;

  rcu_read_lock();
- rss = rcu_dereference(vma->vm_mm->rss_container);
+ rss = rcu_dereference(mm->rss_container);
  css_get(&rss->css);
  rcu_read_unlock();

- pc = kmalloc(sizeof(struct page_container), GFP_KERNEL);
```

```
- if (pc == NULL)
-  goto out_nomem;
+ /* Charge the respective resource count */
+ while (rc) {
+  if (flags & PAGE_TYPE_RSS)
+   rc = res_counter_charge(&rss->res, 1);
+  else
+   rc = res_counter_charge(&rss->pagecache_res, 1);
+
+  if (!rc)
+   break; /* All well */

- while (res_counter_charge(&rss->res, 1)) {
   if (try_to_free_pages_in_container(rss)) {
    atomic_inc(&rss->rss_reclaimed);
-   continue;
+   continue; /* Try agin to charge container */
   }
-
+  /* Unable to free memory?? */
   container_out_of_memory(rss);
   if (test_thread_flag(TIF_MEMDIE))
    goto out_charge;
  }

- pc->page = page;
- pc->cnt = rss;
+  /* Page may have been added to container earlier */
+  pc = page_container(page);
+  if (!pc) {
+   pc = kzalloc(sizeof(struct page_container), GFP_KERNEL);
+   if (pc == NULL)
+    goto out_nomem;
+
+  pc->page = page;
+  pc->cnt = rss;
+  }
+
  *ppc = pc;
  return 0;

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

```
-void container_rss_release(struct page_container *pc)
+void container_page_release(struct page_container *pc, unsigned int flags)
+{
+ struct rss_container *rss;
+
+ rss = pc->cnt;
+ /* Setting the accounts right */
+ if (flags & PAGE_TYPE_RSS) {
+  res_counter_uncharge(&rss->res, 1);
+  if (pc->flags & PAGE_IN_PAGECACHE)
+   res_counter_charge(&rss->pagecache_res, 1);
+ } else {
+  res_counter_uncharge(&rss->pagecache_res, 1);
+ }
+
+ if (!(pc->flags)) {
+  page_container(pc->page) = 0;
+  kfree(pc);
+ }
+ css_put(&rss->css);
+}
+
+void container_page_add(struct page_container *pc, unsigned int flags)
+{
+ struct page *pg;
+ struct rss_container *rss;
+
+ pg = pc->page;
+
+ if (pg == ZERO_PAGE(0))
+  return;
+
+ /* Update flage in page container */
+ if (flags & PAGE_TYPE_RSS) {
+  pc->flags |= PAGE_IN_RSS;
+ } else {
+  pc->flags |= PAGE_IN_PAGECACHE;
+ }
+
+ rss = pc->cnt;
+ spin_lock_irq(&rss->res.lock);
+ if (!pc->ref_cnt)
+  list_add(&pc->list, &rss->inactive_list);
+ pc->ref_cnt++;
+ spin_unlock_irq(&rss->res.lock);
+
+ page_container(pg) = pc;
```

```
+}
+
+void container_page_del(struct page_container *pc, unsigned int flags)
 {
+ struct page *page;
  struct rss_container *rss;

+ page = pc->page;
  rss = pc->cnt;
- res_counter_uncharge(&rss->res, 1);
+
+ if (page == ZERO_PAGE(0))
+  return;
+ BUG_ON(pc->flags & ~3);
+ /* Setting the accounts right */
+ if (flags & PAGE_TYPE_RSS) {
+  res_counter_uncharge(&rss->res, 1);
+  pc->flags &= ~PAGE_IN_RSS;
+  /* If it is a pagecache page the move acct to pagecache */
+  if (pc->flags & PAGE_IN_PAGECACHE)
+   res_counter_charge(&rss->pagecache_res, 1);
+ } else {
+  res_counter_uncharge(&rss->pagecache_res, 1);
+  pc->flags &= ~PAGE_IN_PAGECACHE;
+  BUG_ON(pc->flags);
+ }
+ if(!(pc->flags)) {
+  spin_lock_irq(&rss->res.lock);
+  pc->ref_cnt--;
+  if (!pc->ref_cnt)
+   list_del(&pc->list);
+  spin_unlock_irq(&rss->res.lock);
+
+  if (!pc->ref_cnt) {
+   kfree(pc);
+    page_container(page) = 0;
+  }
+ }
  css_put(&rss->css);
- kfree(pc);
 }

 void container_rss_move_lists(struct page *pg, bool active)
@@ -183,39 +281,6 @@ unsigned long isolate_pages_in_container
  return ret;
 }

-void container_rss_add(struct page_container *pc)
```

```
-{
- struct page *pg;
- struct rss_container *rss;
-
- pg = pc->page;
- rss = pc->cnt;
-
- spin_lock_irq(&rss->res.lock);
- list_add(&pc->list, &rss->active_list);
- spin_unlock_irq(&rss->res.lock);
-
- page_container(pg) = pc;
-}
-
-void container_rss_del(struct page_container *pc)
-{
- struct page *page;
- struct rss_container *rss;
-
- 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);
- init_page_container(page);
-}
-
 static void rss_move_task(struct container_subsys *ss,
   struct container *cont,
   struct container *old_cont,
```