Subject: [RFC][PATCH 2/4] Pagecache accounting
Posted by Vaidyanathan Srinivas on Wed, 20 Jun 2007 11:37:11 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 |   25 ++++--
 mm/rss_container.c            |  171 +++++++++++++++++++++++++++++-----------
 2 files changed, 140 insertions(+), 56 deletions(-)

--- linux-2.6.22-rc2-mm1.orig/include/linux/rss_container.h
+++ linux-2.6.22-rc2-mm1/include/linux/rss_container.h
@@ -12,6 +12,14 @@
 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
 /*
  * This is how the RSS accounting works.
@@ -68,11 +76,12 @@ struct rss_container;
  *
  */

-int container_rss_prepare(struct page *, struct vm_area_struct *vma,
-  struct page_container **);
-void container_rss_add(struct page_container *);
```

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

@@ -92,15 +101,15 @@ static inline int container_rss_prepare(
  return 0;
 }

-static inline void container_rss_add(struct page_container *pc)
+static inline void container_page_add(struct page_container *pc, unsigned int flags)
 {
 }

-static inline void container_rss_del(struct page_container *pc)
+static inline void container_page_del(struct page_container *pc, unsigned int flags)
 {
 }

-static inline void container_rss_release(struct page_container *pc)
+static inline void container_page_release(struct page_container *pc, unsigned int flags)
 {
 }

--- linux-2.6.22-rc2-mm1.orig/mm/rss_container.c
+++ linux-2.6.22-rc2-mm1/mm/rss_container.c
@@ -58,6 +58,8 @@ struct rss_container {
  */

 struct page_container {
+  unsigned long flags;
+  unsigned long ref_cnt;
  struct page *page;
  struct rss_container *cnt;
  struct list_head list; /* this is the element of (int)active_list of
@@ -65,6 +67,11 @@ struct page_container {
     */
 };

+/* 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),
@@ -95,27 +102,39 @@ 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,
-  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;
-
- 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;
-
- while (res_counter_charge(&rss->res, 1)) {
-  if (try_to_free_pages_in_container(rss)) {
-   atomic_inc(&rss->rss_reclaimed);
-   continue;
-  }
+   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;
+ }

+  /* 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 */
+
+    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
    * of reclaim. Some pages are reclaimed and might be moved
@@ -126,61 +145,117 @@ int container_rss_prepare(struct page *p
    */
   if (res_counter_check_under_limit(&rss->res))
    continue;
+  if (res_counter_check_under_limit(&rss->pagecache_res))
+   continue;

-  container_out_of_memory(rss);
-  if (test_thread_flag(TIF_MEMDIE))
-   goto out_charge;
- }
-
- pc->page = page;
- pc->cnt = rss;
- *ppc = pc;
- return 0;
-
-out_charge:
- kfree(pc);
-out_nomem:
- css_put(&rss->css);
- return -ENOMEM;
+   /* 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;
+   }
+
+   *ppc = pc;
+   return 0;
+
+  out_charge:
+  /* 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;
- res_counter_uncharge(&rss->res, 1);
+ /* Setting the accounts right */
+ if (flags & PAGE_TYPE_RSS) {
+  res_counter_uncharge(&rss->res, 1);
+ } else {
+  res_counter_uncharge(&rss->pagecache_res, 1);
+ }
+
+ if (!(pc->flags)) {
+  BUG_ON(pc->ref_cnt);
+  set_page_container(pc->page, NULL);
+  kfree(pc);
+ }
  css_put(&rss->css);
- kfree(pc);
 }

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

  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);
+ 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_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, unsigned int flags)
 {
+ 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);

+ 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 */
+ } else {
+  res_counter_uncharge(&rss->pagecache_res, 1);
```

```
+  pc->flags &= ~PAGE_IN_PAGECACHE;
+ }
+ 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) {
+  BUG_ON(pc->flags);
+  kfree(pc);
+ }
  css_put(&rss->css);
- kfree(pc);
 }

 /*
```