Subject: [RFC][PATCH 2/3] Pagecache and RSS accounting hooks
Posted by Vaidyanathan Srinivas on Fri, 29 Jun 2007 06:21:01 GMT
View Forum Message <> Reply to Message

Pagecache and RSS accounting Hooks

New calls have been added from swap_state.c and filemap.c to track pagecache and swapcache pages.

All existing RSS hooks have been generalised for pagecache accounting as well.

Most of these are function prototype changes.

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

```
 fs/exec.c       |    4 ++--
 mm/filemap.c    |   17 ++++++++++++++++
 mm/memory.c     |   18 +++++++++---------
 mm/migrate.c    |    4 ++--
 mm/rmap.c       |   12 ++++++------
 mm/swap_state.c |   16 ++++++++++++++++
 mm/swapfile.c   |    4 ++--
 7 files changed, 54 insertions(+), 21 deletions(-)

--- linux-2.6.22-rc2-mm1.orig/fs/exec.c
+++ linux-2.6.22-rc2-mm1/fs/exec.c
@@ -321,7 +321,7 @@ void install_arg_page(struct vm_area_str
  if (unlikely(anon_vma_prepare(vma)))
   goto out;

- if (container_rss_prepare(page, vma, &pcont))
+ if (container_page_prepare(page, vma->vm_mm, &pcont))
   goto out;

  flush_dcache_page(page);
@@ -343,7 +343,7 @@ void install_arg_page(struct vm_area_str
  return;

 out_release:
- container_rss_release(pcont);
+ container_page_release(pcont);
 out:
  __free_page(page);
  force_sig(SIGKILL, current);
--- linux-2.6.22-rc2-mm1.orig/mm/filemap.c
+++ linux-2.6.22-rc2-mm1/mm/filemap.c
@@ -30,6 +30,7 @@
```

```
 #include <linux/security.h>
 #include <linux/syscalls.h>
 #include <linux/cpuset.h>
+#include <linux/rss_container.h>
 #include "filemap.h"
 #include "internal.h"

@@ -117,6 +118,9 @@ void __remove_from_page_cache(struct pag
  struct address_space *mapping = page->mapping;

  radix_tree_delete(&mapping->page_tree, page->index);
+ /* Uncharge before the mapping is gone */
+ if (page_container(page))
+  container_page_del(page_container(page));
  page->mapping = NULL;
  mapping->nrpages--;
  __dec_zone_page_state(page, NR_FILE_PAGES);
@@ -440,6 +444,8 @@ int add_to_page_cache(struct page *page,
   pgoff_t offset, gfp_t gfp_mask)
 {
  int error = radix_tree_preload(gfp_mask & ~__GFP_HIGHMEM);
+ struct page_container *pc;
+ struct mm_struct *mm;

  if (error == 0) {
   write_lock_irq(&mapping->tree_lock);
@@ -453,6 +459,17 @@ int add_to_page_cache(struct page *page,
    __inc_zone_page_state(page, NR_FILE_PAGES);
   }
   write_unlock_irq(&mapping->tree_lock);
+  /* Unlock before charge, because we may reclaim this inline */
+  if(!error) {
+   if (current->mm)
+    mm = current->mm;
+   else
+    mm = &init_mm;
+   if (!container_page_prepare(page, mm, &pc))
+    container_page_add(pc);
+   else
+    BUG();
+  }
   radix_tree_preload_end();
  }
  return error;
--- linux-2.6.22-rc2-mm1.orig/mm/memory.c
+++ linux-2.6.22-rc2-mm1/mm/memory.c
@@ -1755,7 +1755,7 @@ gotten:
   cow_user_page(new_page, old_page, address, vma);
```

```
  }

- if (container_rss_prepare(new_page, vma, &pcont))
+ if (container_page_prepare(new_page, vma->vm_mm, &pcont))
    goto oom;

  /*
@@ -1791,7 +1791,7 @@ gotten:
    new_page = old_page;
    ret |= VM_FAULT_WRITE;
  } else
-  container_rss_release(pcont);
+  container_page_release(pcont);

  if (new_page)
    page_cache_release(new_page);
@@ -2217,7 +2217,7 @@ static int do_swap_page(struct mm_struct
    count_vm_event(PGMAJFAULT);
  }

- if (container_rss_prepare(page, vma, &pcont)) {
+ if (container_page_prepare(page, vma->vm_mm, &pcont)) {
    ret = VM_FAULT_OOM;
    goto out;
  }
@@ -2235,7 +2235,7 @@ static int do_swap_page(struct mm_struct

  if (unlikely(!PageUptodate(page))) {
    ret = VM_FAULT_SIGBUS;
-  container_rss_release(pcont);
+  container_page_release(pcont);
    goto out_nomap;
  }

@@ -2271,7 +2271,7 @@ unlock:
 out:
  return ret;
 out_nomap:
- container_rss_release(pcont);
+ container_page_release(pcont);
  pte_unmap_unlock(page_table, ptl);
  unlock_page(page);
  page_cache_release(page);
@@ -2302,7 +2302,7 @@ static int do_anonymous_page(struct mm_s
    if (!page)
      goto oom;

-  if (container_rss_prepare(page, vma, &pcont))
```

```
+  if (container_page_prepare(page, vma->vm_mm, &pcont))
    goto oom_release;

   entry = mk_pte(page, vma->vm_page_prot);
@@ -2338,7 +2338,7 @@ unlock:
  pte_unmap_unlock(page_table, ptl);
  return VM_FAULT_MINOR;
 release_container:
- container_rss_release(pcont);
+ container_page_release(pcont);
 release:
  page_cache_release(page);
  goto unlock;
@@ -2442,7 +2442,7 @@ static int __do_fault(struct mm_struct *

  }

- if (container_rss_prepare(page, vma, &pcont)) {
+ if (container_page_prepare(page, vma->vm_mm, &pcont)) {
   fdata.type = VM_FAULT_OOM;
   goto out;
  }
@@ -2483,7 +2483,7 @@ static int __do_fault(struct mm_struct *
   update_mmu_cache(vma, address, entry);
   lazy_mmu_prot_update(entry);
  } else {
-  container_rss_release(pcont);
+  container_page_release(pcont);
   if (anon)
    page_cache_release(page);
   else
--- linux-2.6.22-rc2-mm1.orig/mm/migrate.c
+++ linux-2.6.22-rc2-mm1/mm/migrate.c
@@ -159,7 +159,7 @@ static void remove_migration_pte(struct
    return;
   }

- if (container_rss_prepare(new, vma, &pcont)) {
+ if (container_page_prepare(new, vma->vm_mm, &pcont)) {
   pte_unmap(ptep);
   return;
  }
@@ -194,7 +194,7 @@ static void remove_migration_pte(struct

 out:
  pte_unmap_unlock(ptep, ptl);
- container_rss_release(pcont);
+ container_page_release(pcont);
```

```
 }

 /*
--- linux-2.6.22-rc2-mm1.orig/mm/rmap.c
+++ linux-2.6.22-rc2-mm1/mm/rmap.c
@@ -578,14 +578,14 @@ void page_add_anon_rmap(struct page *pag
  if (atomic_inc_and_test(&page->_mapcount)) {
   __page_set_anon_rmap(page, vma, address);
   /* 0 -> 1 state change */
-  container_rss_add(pcont);
+  container_page_add(pcont);
  } else {
   __page_check_anon_rmap(page, vma, address);
   /*
    * we raced with another touch or just mapped the page
    * for the N-th time
    */
-  container_rss_release(pcont);
+  container_page_release(pcont);
  }
 }

@@ -606,7 +606,7 @@ void page_add_new_anon_rmap(struct page
 {
  BUG_ON(address < vma->vm_start || address >= vma->vm_end);
  atomic_set(&page->_mapcount, 0); /* elevate count by 1 (starts at -1) */
- container_rss_add(pcont);
+ container_page_add(pcont);
  __page_set_anon_rmap(page, vma, address);
 }

@@ -628,10 +628,10 @@ void page_add_file_rmap(struct page *pag
     * are not added to the container as they do not imply
     * RSS consumption  --xemul
     */
-   container_rss_add(pcont);
+   container_page_add(pcont);
   __inc_zone_page_state(page, NR_FILE_MAPPED);
  } else if (pcont)
-  container_rss_release(pcont);
+  container_page_release(pcont);
 }

 #ifdef CONFIG_DEBUG_VM
@@ -689,7 +689,7 @@ void page_remove_rmap(struct page *page,
   }

   if (pcont)
```

```
-   container_rss_del(pcont);
+   container_page_del(pcont);
    /*
     * It would be tidy to reset the PageAnon mapping here,
     * but that might overwrite a racing page_add_anon_rmap
--- linux-2.6.22-rc2-mm1.orig/mm/swap_state.c
+++ linux-2.6.22-rc2-mm1/mm/swap_state.c
@@ -17,6 +17,7 @@
 #include <linux/backing-dev.h>
 #include <linux/pagevec.h>
 #include <linux/migrate.h>
+#include <linux/rss_container.h>

 #include <asm/pgtable.h>

@@ -74,6 +75,8 @@ static int __add_to_swap_cache(struct pa
          gfp_t gfp_mask)
 {
  int error;
+ struct page_container *pc;
+ struct mm_struct *mm;

  BUG_ON(PageSwapCache(page));
  BUG_ON(PagePrivate(page));
@@ -92,6 +95,17 @@ static int __add_to_swap_cache(struct pa
    }
    write_unlock_irq(&swapper_space.tree_lock);
    radix_tree_preload_end();
+   /* Unlock before charge, because we may reclaim this inline */
+   if(!error) {
+    if (current->mm)
+     mm = current->mm;
+    else
+     mm = &init_mm;
+    if (!container_page_prepare(page, mm, &pc))
+     container_page_add(pc);
+    else
+     BUG();
+   }
   }
   return error;
 }
@@ -130,6 +144,8 @@ void __delete_from_swap_cache(struct pag
   BUG_ON(PagePrivate(page));

   radix_tree_delete(&swapper_space.page_tree, page_private(page));
+ if (page_container(page))
+   container_page_del(page_container(page));
```

```
  set_page_private(page, 0);
  ClearPageSwapCache(page);
  total_swapcache_pages--;
--- linux-2.6.22-rc2-mm1.orig/mm/swapfile.c
+++ linux-2.6.22-rc2-mm1/mm/swapfile.c
@@ -535,7 +535,7 @@ static int unuse_pte_range(struct vm_are
  int found = 0;
  struct page_container *pcont;

- if (container_rss_prepare(page, vma, &pcont))
+ if (container_page_prepare(page, vma->vm_mm, &pcont))
    return 0;

  pte = pte_offset_map_lock(vma->vm_mm, pmd, addr, &ptl);
@@ -552,7 +552,7 @@ static int unuse_pte_range(struct vm_are
  } while (pte++, addr += PAGE_SIZE, addr != end);
  pte_unmap_unlock(pte - 1, ptl);
  if (!found)
-  container_rss_release(pcont);
+  container_page_release(pcont);
  return found;
 }
```