Subject: [PATCH 5/5] blk-throttle: buffered and anonymous page tracking instrumentation
Posted by Andrea Righi on Tue, 22 Feb 2011 17:12:56 GMT
View Forum Message <> Reply to Message

Apply the buffered and anonymous page tracking hooks to the opportune
kernel functions.

Signed-off-by: Ryo Tsuruta <ryov@valinux.co.jp>
Signed-off-by: Hirokazu Takahashi <taka@valinux.co.jp>
Signed-off-by: Andrea Righi <arighi@develer.com>
---
 fs/buffer.c | 1 +
 mm/bounce.c | 1 +
 mm/filemap.c | 1 +
 mm/memory.c | 5 +++++
 mm/page-writeback.c | 1 +
 mm/swap_state.c | 2 ++
 6 files changed, 11 insertions(+), 0 deletions(-)

diff --git a/fs/buffer.c b/fs/buffer.c
index 2219a76..6e473e6 100644
--- a/fs/buffer.c
+++ b/fs/buffer.c
@@ -667,6 +667,7 @@ static void __set_page_dirty(struct page *page,
  if (page->mapping) { /* Race with truncate? */
   WARN_ON_ONCE(warn && !PageUptodate(page));
   account_page_dirtied(page, mapping);
+ blk_throtl_set_filepage_owner(page, current->mm);
   radix_tree_tag_set(&mapping->page_tree,
     page_index(page), PAGECACHE_TAG_DIRTY);
  }
diff --git a/mm/bounce.c b/mm/bounce.c
index 1481de6..f85cafa 100644
--- a/mm/bounce.c
+++ b/mm/bounce.c
@@ -211,6 +211,7 @@ static void __blk_queue_bounce(struct request_queue *q, struct bio
**bio_orig,
   to->bv_len = from->bv_len;
   to->bv_offset = from->bv_offset;
   inc_zone_page_state(to->bv_page, NR_BOUNCE);
+ blk_throtl_copy_page_owner(to->bv_page, page);

   if (rw == WRITE) {
    char *vto, *vfrom;
diff --git a/mm/filemap.c b/mm/filemap.c
index 83a45d3..7fca2b8 100644
--- a/mm/filemap.c

```
+++ b/mm/filemap.c
@@ -407,6 +407,7 @@ int add_to_page_cache_locked(struct page *page, struct address_space
*mapping,
      gfp_mask & GFP_RECLAIM_MASK);
  if (error)
   goto out;
+ blk_throtl_set_filepage_owner(page, current->mm);

  error = radix_tree_preload(gfp_mask & ~__GFP_HIGHMEM);
  if (error == 0) {
diff --git a/mm/memory.c b/mm/memory.c
index 8e8c183..ad5906b 100644
--- a/mm/memory.c
+++ b/mm/memory.c
@@ -52,6 +52,7 @@
 #include <linux/init.h>
 #include <linux/writeback.h>
 #include <linux/memcontrol.h>
+#include <linux/blkdev.h>
 #include <linux/mmu_notifier.h>
 #include <linux/kallsyms.h>
 #include <linux/swapops.h>
@@ -2391,6 +2392,7 @@ gotten:
    */
   ptep_clear_flush(vma, address, page_table);
   page_add_new_anon_rmap(new_page, vma, address);
+  blk_throtl_set_anonpage_owner(new_page, mm);
   /*
    * We call the notify macro here because, when using secondary
    * mmu page tables (such as kvm shadow page tables), we want the
@@ -2826,6 +2828,7 @@ static int do_swap_page(struct mm_struct *mm, struct vm_area_struct
*vma,
  flush_icache_page(vma, page);
  set_pte_at(mm, address, page_table, pte);
  do_page_add_anon_rmap(page, vma, address, exclusive);
+ blk_throtl_set_anonpage_owner(page, mm);
  /* It's better to call commit-charge after rmap is established */
  mem_cgroup_commit_charge_swapin(page, ptr);

@@ -2957,6 +2960,7 @@ static int do_anonymous_page(struct mm_struct *mm, struct
vm_area_struct *vma,

  inc_mm_counter_fast(mm, MM_ANONPAGES);
  page_add_new_anon_rmap(page, vma, address);
+ blk_throtl_set_anonpage_owner(page, mm);
 setpte:
  set_pte_at(mm, address, page_table, entry);
```

```
@@ -3106,6 +3110,7 @@ static int __do_fault(struct mm_struct *mm, struct vm_area_struct
*vma,
   if (anon) {
    inc_mm_counter_fast(mm, MM_ANONPAGES);
    page_add_new_anon_rmap(page, vma, address);
+   blk_throtl_set_anonpage_owner(page, mm);
   } else {
    inc_mm_counter_fast(mm, MM_FILEPAGES);
    page_add_file_rmap(page);
diff --git a/mm/page-writeback.c b/mm/page-writeback.c
index 2cb01f6..277c323 100644
--- a/mm/page-writeback.c
+++ b/mm/page-writeback.c
@@ -1168,6 +1168,7 @@ int __set_page_dirty_nobuffers(struct page *page)
    BUG_ON(mapping2 != mapping);
    WARN_ON_ONCE(!PagePrivate(page) && !PageUptodate(page));
    account_page_dirtied(page, mapping);
+   blk_throtl_set_filepage_owner(page, current->mm);
    radix_tree_tag_set(&mapping->page_tree,
     page_index(page), PAGECACHE_TAG_DIRTY);
   }
diff --git a/mm/swap_state.c b/mm/swap_state.c
index 5c8cfab..bc3a138 100644
--- a/mm/swap_state.c
+++ b/mm/swap_state.c
@@ -16,6 +16,7 @@
 #include <linux/pagemap.h>
 #include <linux/buffer_head.h>
 #include <linux/backing-dev.h>
+#include <linux/blkdev.h>
 #include <linux/pagevec.h>
 #include <linux/migrate.h>
 #include <linux/page_cgroup.h>
@@ -330,6 +331,7 @@ struct page *read_swap_cache_async(swp_entry_t entry, gfp_t
gfp_mask,
   /* May fail (-ENOMEM) if radix-tree node allocation failed. */
   __set_page_locked(new_page);
   SetPageSwapBacked(new_page);
+  blk_throtl_set_anonpage_owner(new_page, current->mm);
   err = __add_to_swap_cache(new_page, entry);
   if (likely(!err)) {
    radix_tree_preload_end();
--
1.7.1
```

_____

Containers mailing list
Containers@lists.linux-foundation.org