

# [PATCH 02/14] page-replace-try\_pageout.patch

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-12/msg08475.html>

---

- *From:* Peter Zijlstra <a.p.zijlstra@xxxxxxxxxx>
  - *Date:* Fri, 30 Dec 2005 23:40:34 +0100
- 

From: Peter Zijlstra <a.p.zijlstra@xxxxxxxxxx>

Move the functionality of the shrink\_list() loop into its own function.

Signed-off-by: Peter Zijlstra <a.p.zijlstra@xxxxxxxxxx>

mm/vmscan.c | 324 +++-----  
1 file changed, 174 insertions(+), 150 deletions(-)

Index: linux-2.6-git/mm/vmscan.c

```
-----  
--- linux-2.6-git.orig/mm/vmscan.c  
+++ linux-2.6-git/mm/vmscan.c  
@@ -51,6 +51,16 @@ typedef enum {  
PAGE_CLEAN,  
} pageout_t;  
  
+/* possible outcome of try_pageout() */  
+typedef enum {  
+ /* unable to perform pageout */  
+ PAGEOUT_KEEP,  
+ /* unable to perform pageout, page is in active use */  
+ PAGEOUT_ACTIVATE,  
+ /* pageout succeeded, page is gone */  
+ PAGEOUT_SUCCESS,  
+} try_pageout_t;  
+  
struct scan_control {  
/* Ask refill_inactive_zone, or shrink_cache to scan this many pages */  
unsigned long nr_to_scan;  
@@ -382,189 +392,203 @@ static pageout_t pageout(struct page *pa  
return PAGE_CLEAN;  
}  
  
-/*  
- * shrink_list adds the number of reclaimed pages to sc->nr_reclaimed  
- */  
-static int shrink_list(struct list_head *page_list, struct scan_control *sc)
```

[PATCH 02/14] page-replace-try\_pageout.patch

```
+static try_pageout_t try_pageout(struct page *page, struct scan_control *sc)
{
- LIST_HEAD(ret_pages);
- struct pagevec freed_pvec;
- int pgactivate = 0;
- int reclaimed = 0;
-
- cond_resched();
-
- pagevec_init(&freed_pvec, 1);
- while (!list_empty(page_list)) {
- struct address_space *mapping;
- struct page *page;
- int may_enter_fs;
- int referenced;
-
- cond_resched();
-
- page = lru_to_page(page_list);
- list_del(&page->lru);
+ struct address_space *mapping;
+ int may_enter_fs;
+ int referenced;

- if (TestSetPageLocked(page))
- goto keep;
+ if (TestSetPageLocked(page))
+ goto keep;

- BUG_ON(PageActive(page));
+ BUG_ON(PageActive(page));

+ sc->nr_scanned++;
+ /* Double the slab pressure for mapped and swapcache pages */
+ if (page_mapped(page) || PageSwapCache(page))
sc->nr_scanned++;
- /* Double the slab pressure for mapped and swapcache pages */
- if (page_mapped(page) || PageSwapCache(page))
- sc->nr_scanned++;

- if (PageWriteback(page))
- goto keep_locked;
+ if (PageWriteback(page))
+ goto keep_locked;

- referenced = page_referenced(page, 1);
- /* In active use or really unfreeable? Activate it. */
- if (referenced && page_mapping_inuse(page))
- goto activate_locked;
+ referenced = page_referenced(page, 1);
+ /* In active use or really unfreeable? Activate it. */
```

```

+ if (referenced && page_mapping_inuse(page))
+ goto activate_locked;

#ifdef CONFIG_SWAP
- /*
- * Anonymous process memory has backing store?
- * Try to allocate it some swap space here.
- */
- if (PageAnon(page) && !PageSwapCache(page)) {
- if (!sc->may_swap)
- goto keep_locked;
- if (!add_to_swap(page))
- goto activate_locked;
- }
+ /*
+ * Anonymous process memory has backing store?
+ * Try to allocate it some swap space here.
+ */
+ if (PageAnon(page) && !PageSwapCache(page)) {
+ if (!sc->may_swap)
+ goto keep_locked;
+ if (!add_to_swap(page))
+ goto activate_locked;
+ }
#endif /* CONFIG_SWAP */

- mapping = page_mapping(page);
- may_enter_fs = (sc->gfp_mask & __GFP_FS) ||
- (PageSwapCache(page) && (sc->gfp_mask & __GFP_IO));
+ mapping = page_mapping(page);
+ may_enter_fs = (sc->gfp_mask & __GFP_FS) ||
+ (PageSwapCache(page) && (sc->gfp_mask & __GFP_IO));

- /*
- * The page is mapped into the page tables of one or more
- * processes. Try to unmap it here.
- */
- if (page_mapped(page) && mapping) {
- switch (try_to_unmap(page)) {
- case SWAP_FAIL:
- goto activate_locked;
- case SWAP_AGAIN:
- goto keep_locked;
- case SWAP_SUCCESS:
- ; /* try to free the page below */
- }
+ /*
+ * The page is mapped into the page tables of one or more
+ * processes. Try to unmap it here.
+ */
+ if (page_mapped(page) && mapping) {

```

```

+ switch (try_to_unmap(page)) {
+ case SWAP_FAIL:
+ goto activate_locked;
+ case SWAP_AGAIN:
+ goto keep_locked;
+ case SWAP_SUCCESS:
+ ; /* try to free the page below */
+ }
+ }

- if (PageDirty(page)) {
- if (referenced)
- goto keep_locked;
- if (!may_enter_fs)
- goto keep_locked;
- if (laptop_mode && !sc->may_writepage)
- goto keep_locked;
+ if (PageDirty(page)) {
+ if (referenced)
+ goto keep_locked;
+ if (!may_enter_fs)
+ goto keep_locked;
+ if (laptop_mode && !sc->may_writepage)
+ goto keep_locked;

- /* Page is dirty, try to write it out here */
- switch(pageout(page, mapping)) {
- case PAGE_KEEP:
- goto keep_locked;
- case PAGE_ACTIVATE:
- goto activate_locked;
- case PAGE_SUCCESS:
- if (PageWriteback(page) || PageDirty(page))
- goto keep;
+ /* Page is dirty, try to write it out here */
+ switch(pageout(page, mapping)) {
+ case PAGE_KEEP:
+ goto keep_locked;
+ case PAGE_ACTIVATE:
+ goto activate_locked;
+ case PAGE_SUCCESS:
+ if (PageWriteback(page) || PageDirty(page))
+ goto keep;
+ /*
+  * A synchronous write - probably a ramdisk. Go
+  * ahead and try to reclaim the page.
+  */
- if (TestSetPageLocked(page))
- goto keep;
- if (PageDirty(page) || PageWriteback(page))
- goto keep_locked;

```

[PATCH 02/14] page-replace-try\_pageout.patch

```
- mapping = page_mapping(page);
- case PAGE_CLEAN:
- ; /* try to free the page below */
- }
+ * A synchronous write – probably a ramdisk. Go
+ * ahead and try to reclaim the page.
+ */
+ if (TestSetPageLocked(page))
+ goto keep;
+ if (PageDirty(page) || PageWriteback(page))
+ goto keep_locked;
+ mapping = page_mapping(page);
+ case PAGE_CLEAN:
+ ; /* try to free the page below */
+ }
+ }

- /*
- * If the page has buffers, try to free the buffer mappings
- * associated with this page. If we succeed we try to free
- * the page as well.
- *
- * We do this even if the page is PageDirty().
- * try_to_release_page() does not perform I/O, but it is
- * possible for a page to have PageDirty set, but it is actually
- * clean (all its buffers are clean). This happens if the
- * buffers were written out directly, with submit_bh(). ext3
- * will do this, as well as the blockdev mapping.
- * try_to_release_page() will discover that cleanness and will
- * drop the buffers and mark the page clean – it can be freed.
- *
- * Rarely, pages can have buffers and no ->mapping. These are
- * the pages which were not successfully invalidated in
- * truncate_complete_page(). We try to drop those buffers here
- * and if that worked, and the page is no longer mapped into
- * process address space (page_count == 1) it can be freed.
- * Otherwise, leave the page on the LRU so it is swappable.
- */
- if (PagePrivate(page)) {
- if (!try_to_release_page(page, sc->gfp_mask))
- goto activate_locked;
- if (!mapping && page_count(page) == 1)
- goto free_it;
- }
+ /*
+ * If the page has buffers, try to free the buffer mappings
+ * associated with this page. If we succeed we try to free
+ * the page as well.
+ *
+ * We do this even if the page is PageDirty().
+ * try_to_release_page() does not perform I/O, but it is
```

[PATCH 02/14] page-replace-try\_pageout.patch

```
+ * possible for a page to have PageDirty set, but it is actually
+ * clean (all its buffers are clean). This happens if the
+ * buffers were written out directly, with submit_bh(). ext3
+ * will do this, as well as the blockdev mapping.
+ * try_to_release_page() will discover that cleanness and will
+ * drop the buffers and mark the page clean – it can be freed.
+ *
+ * Rarely, pages can have buffers and no ->mapping. These are
+ * the pages which were not successfully invalidated in
+ * truncate_complete_page(). We try to drop those buffers here
+ * and if that worked, and the page is no longer mapped into
+ * process address space (page_count == 1) it can be freed.
+ * Otherwise, leave the page on the LRU so it is swappable.
+ */
+ if (PagePrivate(page)) {
+ if (!try_to_release_page(page, sc->gfp_mask))
+ goto activate_locked;
+ if (!mapping && page_count(page) == 1)
+ goto free_it;
+ }

- if (!mapping)
- goto keep_locked; /* truncate got there first */
+ if (!mapping)
+ goto keep_locked; /* truncate got there first */

- write_lock_irq(&mapping->tree_lock);
+ write_lock_irq(&mapping->tree_lock);

- /*
- * The non-racy check for busy page. It is critical to check
- * PageDirty _after_ making sure that the page is freeable and
- * not in use by anybody. (pagecache + us == 2)
- */
- if (unlikely(page_count(page) != 2))
- goto cannot_free;
- smp_rmb();
- if (unlikely(PageDirty(page)))
- goto cannot_free;
+ /*
+ * The non-racy check for busy page. It is critical to check
+ * PageDirty _after_ making sure that the page is freeable and
+ * not in use by anybody. (pagecache + us == 2)
+ */
+ if (unlikely(page_count(page) != 2))
+ goto cannot_free;
+ smp_rmb();
+ if (unlikely(PageDirty(page)))
+ goto cannot_free;

#ifdef CONFIG_SWAP
```

[PATCH 02/14] page-replace-try\_pageout.patch

```
- if (PageSwapCache(page)) {
- swp_entry_t swap = { .val = page_private(page) };
- __delete_from_swap_cache(page);
- write_unlock_irq(&mapping->tree_lock);
- swap_free(swap);
- __put_page(page); /* The pagecache ref */
- goto free_it;
- }
+ if (PageSwapCache(page)) {
+ swp_entry_t swap = { .val = page_private(page) };
+ __delete_from_swap_cache(page);
+ write_unlock_irq(&mapping->tree_lock);
+ swap_free(swap);
+ __put_page(page); /* The pagecache ref */
+ goto free_it;
+ }
#endif /* CONFIG_SWAP */

- __remove_from_page_cache(page);
- write_unlock_irq(&mapping->tree_lock);
- __put_page(page);
+ __remove_from_page_cache(page);
+ write_unlock_irq(&mapping->tree_lock);
+ __put_page(page);

free_it:
- unlock_page(page);
- reclaimed++;
- if (!pagevec_add(&freed_pvec, page))
- __pagevec_release_nonlru(&freed_pvec);
- continue;
+ unlock_page(page);
+ return PAGEOUT_SUCCESS;

cannot_free:
- write_unlock_irq(&mapping->tree_lock);
- goto keep_locked;
+ write_unlock_irq(&mapping->tree_lock);
+ goto keep_locked;

activate_locked:
- SetPageActive(page);
- pgactivate++;
+ unlock_page(page);
+ return PAGEOUT_ACTIVATE;
keep_locked:
- unlock_page(page);
+ unlock_page(page);
keep:
- list_add(&page->lru, &ret_pages);
- BUG_ON(PageLRU(page));
```



[PATCH 02/14] page-replace-try\_pageout.patch

- Prev by Date: [\*\[PATCH 01/14\] page-replace-single-batch-insert.patch\*](#)
- Next by Date: [\*\[PATCH 10/14\] page-replace-remove-mm inline.patch\*](#)
- Previous by thread: [\*Re: \[PATCH 01/14\] page-replace-single-batch-insert.patch\*](#)
- Next by thread: [\*\[PATCH 10/14\] page-replace-remove-mm inline.patch\*](#)
- Index(es):
  - ◆ [\*Date\*](#)
  - ◆ [\*Thread\*](#)