Subject: [PATCH 2/6 mm] memcgroup: temporarily revert swapoff mod
Posted by Hugh Dickins on Fri, 09 Nov 2007 07:10:23 GMT
View Forum Message <> Reply to Message

Whaaa? This patch precisely reverts the "swapoff: scan ptes preemptibly" patch just presented. It's a temporary measure to allow existing memory controller patches to apply without rejects: in due course they should be rendered down into one sensible patch, and this reversion disappear.

Signed-off-by: Hugh Dickins <hugh@veritas.com>

This patch should go immediately before the memory-controller patches,
or immediately before memory-controller-memory-accounting-v7.patch

mm/swapfile.c | 38 +++++++----------------------------
1 file changed, 7 insertions(+), 31 deletions(-)

```
--- patch1/mm/swapfile.c 2007-11-08 12:34:12.000000000 +0000
+++ patch2/mm/swapfile.c 2007-11-08 12:34:12.000000000 +0000
@@ -506,19 +506,9 @@ unsigned int count_swap_pages(int type,
  * just let do_wp_page work it out if a write is requested later - to
  * force COW, vm_page_prot omits write permission from any private vma.
  */
-static int unuse_pte(struct vm_area_struct *vma, pmd_t *pmd,
+static void unuse_pte(struct vm_area_struct *vma, pte_t *pte,
   unsigned long addr, swp_entry_t entry, struct page *page)
 {
- spinlock_t *ptl;
- pte_t *pte;
- int found = 1;
-
- pte = pte_offset_map_lock(vma->vm_mm, pmd, addr, &ptl);
- if (unlikely(!pte_same(*pte, swp_entry_to_pte(entry)))) {
-  found = 0;
-  goto out;
- }
-
  inc_mm_counter(vma->vm_mm, anon_rss);
  get_page(page);
  set_pte_at(vma->vm_mm, addr, pte,
@@ -530,9 +520,6 @@ static int unuse_pte(struct vm_area_stru
   * immediately swapped out again after swapon.
   */
  activate_page(page);
-out:
- pte_unmap_unlock(pte, ptl);
- return found;
 }
```

```
 static int unuse_pte_range(struct vm_area_struct *vma, pmd_t *pmd,
@@ -541,33 +528,22 @@ static int unuse_pte_range(struct vm_are
 {
  pte_t swp_pte = swp_entry_to_pte(entry);
  pte_t *pte;
+ spinlock_t *ptl;
  int found = 0;

- /*
-  * We don't actually need pte lock while scanning for swp_pte: since
-  * we hold page lock and mmap_sem, swp_pte cannot be inserted into the
-  * page table while we're scanning; though it could get zapped, and on
-  * some architectures (e.g. x86_32 with PAE) we might catch a glimpse
-  * of unmatched parts which look like swp_pte, so unuse_pte must
-  * recheck under pte lock.  Scanning without pte lock lets it be
-  * preemptible whenever CONFIG_PREEMPT but not CONFIG_HIGHPTE.
-  */
- pte = pte_offset_map(pmd, addr);
+ pte = pte_offset_map_lock(vma->vm_mm, pmd, addr, &ptl);
  do {
   /*
    * swapoff spends a _lot_ of time in this loop!
    * Test inline before going to call unuse_pte.
    */
   if (unlikely(pte_same(*pte, swp_pte))) {
-   pte_unmap(pte);
-   found = unuse_pte(vma, pmd, addr, entry, page);
-   if (found)
-    goto out;
-   pte = pte_offset_map(pmd, addr);
+   unuse_pte(vma, pte++, addr, entry, page);
+   found = 1;
+   break;
   }
  } while (pte++, addr += PAGE_SIZE, addr != end);
- pte_unmap(pte - 1);
-out:
+ pte_unmap_unlock(pte - 1, ptl);
  return found;
 }
```