Subject: Re: [RFC] [-mm PATCH] Memory controller fix swap charging context in unuse_pte()
Posted by Balbir Singh on Mon, 15 Oct 2007 17:27:14 GMT
View Forum Message <> Reply to Message

Hugh Dickins wrote:
>
> --- 2.6.23-rc8-mm2/mm/swapfile.c 2007-09-27 12:03:36.000000000 +0100
> +++ linux/mm/swapfile.c 2007-10-07 14:33:05.000000000 +0100
> @@ -507,11 +507,23 @@ 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, pte_t *pte,
> +static int unuse_pte(struct vm_area_struct *vma, pmd_t *pmd,
>  unsigned long addr, swp_entry_t entry, struct page *page)
> {
> + spinlock_t *ptl;
> + pte_t *pte;
> + int ret = 1;
> +
>  if (mem_cgroup_charge(page, vma->vm_mm, GFP_KERNEL))
> - return -ENOMEM;
> + ret = -ENOMEM;
> +
> + pte = pte_offset_map_lock(vma->vm_mm, pmd, addr, &ptl);
> + if (unlikely(!pte_same(*pte, swp_entry_to_pte(entry)))) {
> + if (ret > 0)
> + mem_cgroup_uncharge_page(page);
> + ret = 0;
> + goto out;
> + }
>
>  inc_mm_counter(vma->vm_mm, anon_rss);
>  get_page(page);
> @@ -524,7 +536,9 @@ static int unuse_pte(struct vm_area_stru
>  * immediately swapped out again after swapon.
>  */
>  activate_page(page);
> - return 1;
> +out:
> + pte_unmap_unlock(pte, ptl);
> + return ret;
> }
>
> static int unuse_pte_range(struct vm_area_struct *vma, pmd_t *pmd,
> @@ -533,21 +547,33 @@ static int unuse_pte_range(struct vm_are
> {

```
>   pte_t swp_pte = swp_entry_to_pte(entry);
>   pte_t *pte;
> - spinlock_t *ptl;
>   int ret = 0;
>
> - pte = pte_offset_map_lock(vma->vm_mm, pmd, addr, &ptl);
> + /*
> +  * We don't actually need pte lock while scanning for swp_pte:
> +  * since we hold page lock, swp_pte cannot be inserted into or
> +  * removed from a page table while we're scanning; but on some
> +  * architectures (e.g. i386 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 the lock
> +  * is preemptible if CONFIG_PREEMPT without CONFIG_HIGHPTE.
> +  */
> + pte = pte_offset_map(pmd, addr);
>   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))) {
> -   ret = unuse_pte(vma, pte++, addr, entry, page);
> -   break;
> +   pte_unmap(pte);
> +   ret = unuse_pte(vma, pmd, addr, entry, page);
> +   if (ret)
> +    goto out;
> +   pte = pte_offset_map(pmd, addr);
>    }
>   } while (pte++, addr += PAGE_SIZE, addr != end);
> - pte_unmap_unlock(pte - 1, ptl);
> + pte_unmap(pte - 1);
> +out:
>   return ret;
>  }
>
```

I tested this patch and it seems to be working fine. I tried swapoff -a
in the middle of tests consuming swap. Not 100% rigorous, but a good
test nevertheless.

Tested-by: Balbir Singh <balbir@linux.vnet.ibm.com>

--
Warm Regards,
Balbir Singh
Linux Technology Center

IBM, ISTL

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers