Subject: /proc/*/pagemap BUG: sleeping function called from invalid context
Posted by Alexey Dobriyan on Mon, 09 Apr 2007 08:18:06 GMT
View Forum Message <> Reply to Message

After
 cat /proc/self/pagemap

BUG: sleeping function called from invalid context at include/asm/uaccess.h:453
in_atomic():1, irqs_disabled():0
1 lock held by cat/14183:
 #0: (&mm->mmap_sem){----}, at: [<c017d17b>] pagemap_read+0x11f/0x21b
 [<c01b7bc7>] copy_to_user+0x37/0x4c
 [<c017cf92>] add_to_pagemap+0x49/0x6f
 [<c017d034>] pagemap_pte_range+0x56/0x7e
 [<c017cfde>] pagemap_pte_range+0x0/0x7e
 [<c01b481d>] walk_page_range+0xf1/0x1a0
 [<c017d1e3>] pagemap_read+0x187/0x21b
 [<c01fdafa>] tty_write+0x1bb/0x1cc
 [<c017d05c>] pagemap_read+0x0/0x21b
 [<c01589b0>] vfs_read+0x72/0x95
 [<c0158cce>] sys_read+0x41/0x67
 [<c0103d96>] sysenter_past_esp+0x8f/0x99
 [<c0103d66>] sysenter_past_esp+0x5f/0x99
 =======================

Subject: Re: /proc/*/pagemap BUG: sleeping function called from invalid context
Posted by Matt Mackall on Mon, 09 Apr 2007 17:43:02 GMT
View Forum Message <> Reply to Message

On Mon, Apr 09, 2007 at 12:25:54PM +0400, Alexey Dobriyan wrote:
> After
> cat /proc/self/pagemap
>
> BUG: sleeping function called from invalid context at include/asm/uaccess.h:453
> in_atomic():1, irqs_disabled():0
> 1 lock held by cat/14183:
> #0: (&mm->mmap_sem){----}, at: [<c017d17b>] pagemap_read+0x11f/0x21b
> [<c01b7bc7>] copy_to_user+0x37/0x4c
> [<c017cf92>] add_to_pagemap+0x49/0x6f
> [<c017d034>] pagemap_pte_range+0x56/0x7e
> [<c017cfde>] pagemap_pte_range+0x0/0x7e
> [<c01b481d>] walk_page_range+0xf1/0x1a0
> [<c017d1e3>] pagemap_read+0x187/0x21b
> [<c01fdafa>] tty_write+0x1bb/0x1cc
> [<c017d05c>] pagemap_read+0x0/0x21b
> [<c01589b0>] vfs_read+0x72/0x95
> [<c0158cce>] sys_read+0x41/0x67

> [<c0103d96>] sysenter_past_esp+0x8f/0x99
> [<c0103d66>] sysenter_past_esp+0x5f/0x99

I'm still waking up but this is odd. mm->mmap_sem is an rwsem,
sleeping inside of it should be perfectly valid.

Oh. You've got CONFIG_HIGHPTE and the note is really about the
kmap_atomic inside pte_offset_map. I'll fix that up.

--
Mathematics is the supreme nostalgia of our time.

Subject: Re: /proc/*/pagemap BUG: sleeping function called from invalid context
Posted by Matt Mackall on Mon, 09 Apr 2007 21:43:19 GMT
View Forum Message <> Reply to Message

On Mon, Apr 09, 2007 at 12:25:54PM +0400, Alexey Dobriyan wrote:
> After
> cat /proc/self/pagemap
>
> BUG: sleeping function called from invalid context at include/asm/uaccess.h:453
> in_atomic():1, irqs_disabled():0
> 1 lock held by cat/14183:
> #0: (&mm->mmap_sem){----}, at: [<c017d17b>] pagemap_read+0x11f/0x21b
> [<c01b7bc7>] copy_to_user+0x37/0x4c
> [<c017cf92>] add_to_pagemap+0x49/0x6f
> [<c017d034>] pagemap_pte_range+0x56/0x7e
> [<c017cfde>] pagemap_pte_range+0x0/0x7e
> [<c01b481d>] walk_page_range+0xf1/0x1a0
> [<c017d1e3>] pagemap_read+0x187/0x21b
> [<c01fdafa>] tty_write+0x1bb/0x1cc
> [<c017d05c>] pagemap_read+0x0/0x21b
> [<c01589b0>] vfs_read+0x72/0x95
> [<c0158cce>] sys_read+0x41/0x67
> [<c0103d96>] sysenter_past_esp+0x8f/0x99
> [<c0103d66>] sysenter_past_esp+0x5f/0x99
> =======================

This looks like it was caused by CONFIG_HIGHPTE making pte_offset_map
call kmap_atomic. The message about mm->mmap_sem is spurious. Looking
at a fix.

--
Mathematics is the supreme nostalgia of our time.

Subject: Re: /proc/*/pagemap BUG: sleeping function called from invalid context
Posted by Matt Mackall on Tue, 10 Apr 2007 20:07:03 GMT
View Forum Message <> Reply to Message

On Mon, Apr 09, 2007 at 12:25:54PM +0400, Alexey Dobriyan wrote:
> After
> cat /proc/self/pagemap
>
> BUG: sleeping function called from invalid context at include/asm/uaccess.h:453
> in_atomic():1, irqs_disabled():0

This should fix it:

When CONFIG_HIGHPTE is enabled, use double-buffering in pagemap to
avoid calling copy_to_user while preemption is disabled.

Tested on x86 with HIGHPTE with DEBUG_SPINLOCK_SLEEP and
PROVE_LOCKING.

Signed-off-by: Matt Mackall <mpm@selenic.com>

Index: mm/fs/proc/task_mmu.c
 === =======
--- mm.orig/fs/proc/task_mmu.c 2007-04-09 10:54:28.000000000 -0500
+++ mm/fs/proc/task_mmu.c 2007-04-10 14:47:21.000000000 -0500
@@ -535,6 +535,7 @@ struct pagemapread {
  struct mm_struct *mm;
  unsigned long next;
  unsigned long *buf;
+ pte_t *ptebuf;
  unsigned long pos;
  size_t count;
  int index;
@@ -573,6 +574,14 @@ static int pagemap_pte_range(pmd_t *pmd,
  int err;

  pte = pte_offset_map(pmd, addr);
+
+#ifdef CONFIG_HIGHPTE
+ /* copy PTE directory to temporary buffer and unmap it */
+ memcpy(pm->ptebuf, pte, PAGE_ALIGN((unsigned long)pte) - (unsigned long)pte);
+ pte_unmap(pte);
+ pte = pm->ptebuf;
+#endif
+
  for (; addr != end; pte++, addr += PAGE_SIZE) {
   if (addr < pm->next)
    continue;
@@ -583,7 +592,11 @@ static int pagemap_pte_range(pmd_t *pmd,

```
   if (err)
    return err;
  }
+
+#ifndef CONFIG_HIGHPTE
  pte_unmap(pte - 1);
+#endif
+
  return 0;
 }

@@ -655,10 +668,16 @@ static ssize_t pagemap_read(struct file
  if (!page)
   goto out;

+#ifdef CONFIG_HIGHPTE
+ pm.ptebuf = kzalloc(PAGE_SIZE, GFP_USER);
+ if (!pm.ptebuf)
+  goto out_free;
+#endif
+
  ret = 0;
  mm = get_task_mm(task);
  if (!mm)
-  goto out_free;
+  goto out_freepte;

  pm.mm = mm;
  pm.next = addr;
@@ -681,7 +700,7 @@ static ssize_t pagemap_read(struct file
  while (pm.count > 0 && vma) {
   if (!ptrace_may_attach(task)) {
    ret = -EIO;
-   goto out;
+   goto out_mm;
   }
   vend = min(vma->vm_start - 1, end - 1) + 1;
   ret = pagemap_fill(&pm, vend);
@@ -700,8 +719,13 @@ static ssize_t pagemap_read(struct file
  if (!ret)
   ret = pm.pos - src;

+out_mm:
  mmput(mm);
+out_freepte:
+#ifdef CONFIG_HIGHPTE
+ kfree(pm.ptebuf);
 out_free:
```

```
+#endif
  kfree(page);
 out:
  put_task_struct(task);
```

--
Mathematics is the supreme nostalgia of our time.