

Re: [PATCH] mm: fix page\_mkclean\_one (was: 2.6.19 file content corruption on ext3)

## Re: [PATCH] mm: fix page\_mkclean\_one (was: 2.6.19 file content corruption on ext3)

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-12/msg06189.html>

---

- *From:* Linus Torvalds <torvalds@xxxxxxxx>
  - *Date:* Wed, 20 Dec 2006 11:01:13 -0800 (PST)
- 

On Wed, 20 Dec 2006, Martin Michlmayr wrote:

Anyway, the page\_mkclean\_one() fixes (along with \_most\_ things we've looked at) shouldn't matter on UP, at least certainly not without PREEMPT.

Hmm. So what about UP without PREEMPT then...

So that's why I've been harping on the fact that I think we simply do really wrong things with PG\_dirty at times, and that I find it confusing that there's

– clear\_page\_dirty\_for\_io(): this one makes sense. The name makes sense, and the implementation makes sense (which is \_not\_ the same thing as "works", of course – "makes sense" does not mean "no bugs" ;).

– test\_clear\_page\_dirty: this one makes no sense WHATSOEVER, except as a buggy way to do the "\_for\_io()" case.. This makes sense neither from a concept angle \_or\_ an implementation angle (the whole "test\_" part is nonsense: why would anybody care? What operation does this? What can it do if the page is dirty? It also has no sensible thing it can do to the page tables.

– clear\_page\_dirty(): this one makes sense only as a "cancel" operation, for vmtruncate and friends (it's different from the "\_for\_io()" case in several ways:

(a) we should have unmapped such pages forcibly \_anyway\_, so looking at the PTE's make no sense.

(b) because we're not starting IO, we don't have the "mark for writeback" case, and we need to clear the dirty tags from the radix trees etc since the writeback logic won't do it for us.

The \_implementation\_ of "clear\_page\_dirty()" doesn't make sense, but

Re: [PATCH] mm: fix page\_mkclean\_one (was: 2.6.19 file content corruption on ext3)

Re: [PATCH] mm: fix page\_mkclean\_one (was: 2.6.19 file content corruption on ext3)

the concept does.

I've repeated that theory a few times, but neither Andrew nor Nick seem to really believe in it. So I'll just repeat it once more, only to be shot down. I think we have three operations, one of which is totally idiotic and senseless, and one of which is just badly implemented.

Maybe the following information is helpful in some way: remember how I said that we have applied 6 mm patches to 2.6.18 in Debian? According to Gordon Farquharson, who's helping me a great deal with testing installation on this ARM machine (Linksys NSLU2), the corruption doesn't always show up when you only apply mm-tracking-shared-dirty-pages.patch to 2.6.18 but it shows up all the time with all six patches applied.

I think the "it happens occasionally with just the first patch" is the really important part. The other patches really are likely to just change writeback timing behaviour (\_especially\_ the "tracking-shared-dirty-pages" patch), but if it happens occasionally even with the first one, that's the one that almost certainly introduced the real problem.

And my argument above is actually that the "real problem" goes a hell of a lot further back in time, but it didn't use to be a problem because we just considered dirty bits in the page tables to be something \_completely\_ independent of the "page dirty" status, so historically, it just didn't matter that we had insane implementations and senseless operations.

Linus

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>