DragonFlyBSD - Bug #1917

panic: assertion: (RB_EMPTY(&ip->rec_tree) && (ip->flags & HAMMER_INODE_XDIRTY) == 0) || (!RB_EMPTY(&ip->rec_tree) && (ip->flags & HAMMER_INODE_XDIRTY) != 0) in hammer_flush_inode_done

11/21/2010 11:23 AM - qhwt.dfly

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Hello.
Caught this while running pbulk on an Atom D510, running x86_64 kernel.
This one is new to me and I haven't seen this one before updating the
kernel. The current kernel was built from source whose commit hash is
c6aff4f7f0c74961380763de0bcb5dd49fb7faea, while the previous kernel was
from c4ce4f88f289f77826fecf470965d7e216274018. No hammer cleanup was
running when it panicked.

```
Unread portion of the kernel message buffer:
1548 1532 1516 1500 1484 1468SECONDARY PANIC ON CPU 3 THREAD 0xffffffe02c36cc70
SECONDARY PANIC ON CPU 1 THREAD 0xffffffe02c36c670
SECONDARY PANIC ON CPU 2 THREAD 0xffffffe0526d7a70
1452SECONDARY PANIC ON CPU 2 THREAD 0xffffffe052996870
SECONDARY PANIC ON CPU 2 THREAD 0xffffffe0526d6870
SECONDARY PANIC ON CPU 3 THREAD 0xffffffe052bac070
SECONDARY PANIC ON CPU 3 THREAD 0xffffffe0526dd470
1436SECONDARY PANIC ON CPU 3 THREAD 0xffffffe02c369070
1420 1404 1388 1372 1356 1340 1324 1308 1292 1276 1260 1244 1228 1212 1196 1180 1164 1148 1132 1116 1100 1084 1068
1052 1036 1020 1004 988 972 956 940 924 908 892 876 860 844 828 812 796 780 764 748 732 716 700 684 668 652 636 620 604
588 572 556 540 524 508 492 476 460 444 428 412 396 380 364 348 332 316 300 284 268 252 236 220 204 188 172 156 140 124
108 92 76 60 44 28 12

Reading symbols from /boot/kernel/acpi.ko...done.
Loaded symbols for /boot/kernel/acpi.ko
Reading symbols from /boot/kernel/ahci.ko...done.
Loaded symbols for /boot/kernel/ahci.ko
Reading symbols from /boot/kernel/ehci.ko...done.
Loaded symbols for /boot/kernel/ehci.ko
_get_mycpu (di=0xffffffff80719020) at ./machine/thread.h:73
73          __asm ("movq %%gs:globaldata,%0" : "=r" (gd) : "m"(__mycpu__dummy));
(kgdb) bt
#0  _get_mycpu (di=0xffffffff80719020) at ./machine/thread.h:73
#1  md_dumpsys (di=0xffffffff80719020)
at /usr/src/sys/platform/pc64/x86_64/dump_machdep.c:262
#2  0xffffffff802a5b6c in dumpsys () at /usr/src/sys/kern/kern_shutdown.c:881
#3  0xffffffff802a62e9 in boot (howto=-2004318071)
at /usr/src/sys/kern/kern_shutdown.c:388
#4  0xffffffff802a66a7 in panic (fmt=0xffffffff804d4f2a "assertion: %s in %s")
at /usr/src/sys/kern/kern_shutdown.c:787
#5  0xffffffff804030f8 in hammer_flush_inode_done (ip=0xffffffe06fc17a78,
error=<value optimized out>) at /usr/src/sys/vfs/hammer/hammer_inode.c:2421
#6  0xffffffff803ff587 in hammer_flusher_flush_inode (
arg=<value optimized out>) at /usr/src/sys/vfs/hammer/hammer_flusher.c:525
#7  hammer_flusher_slave_thread (arg=<value optimized out>)
at /usr/src/sys/vfs/hammer/hammer_flusher.c:455
#8  0xffffffff802b1a2a in lwkt_deschedule_self (td=0x0)
at /usr/src/sys/kern/lwkt_thread.c:258
Backtrace stopped: previous frame inner to this frame (corrupt stack?)
(kgdb) fr 5
```

```
#5  0xffffffff804030f8 in hammer_flush_inode_done (ip=0xffffffe06fc17a78,
error=<value optimized out>) at /usr/src/sys/vfs/hammer/hammer_inode.c:2421
2421            KKASSERT((RB_EMPTY(&ip->rec_tree) &&
(kgdb) p ip->rec_tree
$1 = {rbh_root = 0x0, rbh_inprog = 0x0}
(kgdb) p/x ip->flags
$2 = 0x1353813
(kgdb)
```

I'm preparing to upload the kern.18 and vmcore.18.

History

#1 - 11/21/2010 07:54 PM - dillon

:Hello.
:Caught this while running pbulk on an Atom D510, running x86_64 kernel.
:This one is new to me and I haven't seen this one before updating the
:kernel. The current kernel was built from source whose commit hash is
:c6aff4f7f0c74961380763de0bcb5dd49fb7faea, while the previous kernel was
:from c4ce4f88f289f77826fecf470965d7e216274018. No hammer cleanup was
:running when it panicked.

Hmm. Well, the state of the tree and flags as-of the core dump
is fine and would have passed that assertion test, so something
changed between the assertion and the kernel core dump.

Essentially that assertion is saying that XDIRTY should always be
in a synchronized state against whether records exist in ip->rec_tree
or not.

I think you may have found a very short race condition where the
XDIRTY flag is not perfectly in-sync with the presence of records
on the RB tree, but I can only guess. It's going to be very rare.

I found one situation where this can happen. If the frontend truncates
a file and calls hammer_rel_mem_record() simultaniously with the backend
finishing a flush operation on the same inode AND a direct write I/O
is in progress (causing the frontend to block) then this case can occur.

Here is a patch to try:

fetch http://apollo.backplane.com/DFlyMisc/hammer25.patch

However it may be very difficult to reproduce this bug. What we really
need to do here is just do a run-through on the HAMMER filesystem w/
the patch to make sure I didn't break anything by moving the clearing
of those flags. I'm pretty sure it is ok but I get a bit jittery when
I have to mess with hammer_inode->flags.

-Matt

#2 - 11/24/2010 12:19 AM - qhwt.dfly

Hi.

On Sun, Nov 21, 2010 at 11:53:30AM -0800, Matthew Dillon wrote:
>   Here is a patch to try:
>
> fetch http://apollo.backplane.com/DFlyMisc/hammer25.patch
>
>   However it may be very difficult to reproduce this bug. What we really
>   need to do here is just do a run-through on the HAMMER filesystem w/
>   the patch to make sure I didn't break anything by moving the clearing
>   of those flags. I'm pretty sure it is ok but I get a bit jittery when
>   I have to mess with hammer_inode->flags.

I've applied the patch and re-installed the kernel, and restarted the
bulkbuild. 48 hours have past since then and so far I've observed no
panic or any other odd things. Is this enough for a run-through?

Thanks.

#3 - 11/24/2010 03:23 AM - dillon

:I've applied the patch and re-installed the kernel, and restarted the
:bulkbuild. 48 hours have past since then and so far I've observed no
:panic or any other odd things. Is this enough for a run-through?
:
:Thanks.

Well, probably not enough of a test but a very good start!

I guess the real question is whether it actually fixed the bug
or not. The only way to really tell is the give it a nice long
run with the bug fix and then take out the bug fix and see if
you get the panic again.

It's fairly rare so we might not be able to validate the bug fix,
but I'll commit it anyway once enough testing has been done.
I'm running buildworld and other tests here with the fix as well.
I'll commit the fix in a day or three.

-Matt
Matthew Dillon
<dillon@backplane.com>