DragonFlyBSD - Bug #1671

panic: unmount: dangling vnode

02/09/2010 12:27 AM - rumcic

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

Description

I had a ntfs read-only mount and copied a few files off of it. After that I
tried unmounting it, which resulted in the panic mentioned in the subject. The
core is available at leaf:~rumko/crash/vnode/*.6 and the kernel was from about
31st Jan.

Unread portion of the kernel message buffer:
panic: unmount: dangling vnode
mp_lock = 00000000; cpuid = 0
Trace beginning at frame 0xf32f1c24
panic(f32f1c48,f32f1c58,d9e6bdd8,0,f32f1c6c) at panic+0x14d
panic(c0440231,0,d9e6bdf8,1,10001001) at panic+0x14d
dounmount(d9e6bdd8,0,0,0,0) at dounmount+0x2da
sys_unmount(f32f1cf0,6,206d1,0,c0496228) at sys_unmount+0x13c
syscall2(f32f1d40) at syscall2+0x3ac
Xint0x80_syscall() at Xint0x80_syscall+0x36
boot() called on cpu#0
Uptime: 7d18h39m25s
Physical memory: 3065 MB
Dumping 344 MB: 329 313 297 281 265 249 233 217 201 185 169 153 137 121 105 89
73 57 41 25 9

Reading symbols from /boot/modules/if_nfe.ko...done.
Loaded symbols for /boot/modules/if_nfe.ko
Reading symbols from /boot/modules/powernow.ko...done.
Loaded symbols for /boot/modules/powernow.ko
Reading symbols from /boot/modules/linprocfs.ko...done.
Loaded symbols for /boot/modules/linprocfs.ko
Reading symbols from /boot/modules/if_bridge.ko...done.
Loaded symbols for /boot/modules/if_bridge.ko
Reading symbols from /boot/modules/linux.ko...done.
Loaded symbols for /boot/modules/linux.ko
Reading symbols from /boot/modules/ntfs.ko...done.
Loaded symbols for /boot/modules/ntfs.ko
_get_mycpu (di=0xc04d9de0) at ./machine/thread.h:83
83 __asm ("movl %%fs:globaldata,%0" : "=r"
(gd) : "m"(__mycpu__dummy));
(kgdb) bt
#0 _get_mycpu (di=0xc04d9de0) at ./machine/thread.h:83
#1 md_dumpsys (di=0xc04d9de0)
at /usr/src/sys/platform/pc32/i386/dump_machdep.c:264
#2 0xc01e45c2 in dumpsys () at /usr/src/sys/kern/kern_shutdown.c:838
#3 0xc01e4b94 in boot (howto=260) at /usr/src/sys/kern/kern_shutdown.c:387
#4 0xc01e50f2 in panic (fmt=0xc0440231 "unmount: dangling vnode")
at /usr/src/sys/kern/kern_shutdown.c:744
#5 0xc02449f4 in dounmount (mp=0xd9e6bdd8, flags=0)
at /usr/src/sys/kern/vfs_syscalls.c:783
#6 0xc0244ca0 in sys_unmount (uap=0xf32f1cf0)
at /usr/src/sys/kern/vfs_syscalls.c:618
#7 0xc0403b90 in syscall2 (frame=0xf32f1d40)
at /usr/src/sys/platform/pc32/i386/trap.c:1366
#8 0xc03ef4b6 in Xint0x80_syscall ()
at /usr/src/sys/platform/pc32/i386/exception.s:876

#9 0x0000001f in ?? ()
Backtrace stopped: previous frame inner to this frame (corrupt stack?)
--
Regards,
Rumko

History

#1 - 02/10/2010 02:29 AM - dillon

:I had a ntfs read-only mount and copied a few files off of it. After that I
:tried unmounting it, which resulted in the panic mentioned in the subject. The
:core is available at leaf:~rumko/crash/vnode/*.6 and the kernel was from about
:31st Jan.

Ok, try this patch. It looks like NTFS accesses auxillary vnodes
while flushing other vnodes, so a single vflush scan won't catch them
all. I don't even know if multiple vflushes will catch them all
but lets try it and find out.

-Matt

```
diff --git a/sys/vfs/ntfs/ntfs_vfsops.c b/sys/vfs/ntfs/ntfs_vfsops.c
index d3d80d2..2a8c1a5 100644
--- a/sys/vfs/ntfs/ntfs_vfsops.c
+++ b/sys/vfs/ntfs/ntfs_vfsops.c
@@ -633,6 +633,8 @@ ntfs_unmount(struct mount *mp, int mntflags)

dprintf(("ntfs_unmount: vflushing...\n"));
error = vflush(mp, 0, flags | SKIPSYSTEM);
+ error = vflush(mp, 0, flags | SKIPSYSTEM);
+ error = vflush(mp, 0, flags | SKIPSYSTEM);
if (error) {
kprintf("ntfs_unmount: vflush failed: %d\n",error);
return (error);
@@ -649,6 +651,8 @@ ntfs_unmount(struct mount *mp, int mntflags)

/* vflush system vnodes */
error = vflush(mp, 0, flags);
+ error = vflush(mp, 0, flags);
+ error = vflush(mp, 0, flags);
if (error)
kprintf("ntfs_unmount: vflush failed(sysnodes): %d\n",error);
```

#2 - 02/10/2010 04:57 AM - tbisson

On Feb 9, 2010, at 6:21 PM, Matthew Dillon wrote:

>
> :I had a ntfs read-only mount and copied a few files off of it.
> After that I
> :tried unmounting it, which resulted in the panic mentioned in the
> subject. The
> :core is available at leaf:~rumko/crash/vnode/*.6 and the kernel was
> from about
> :31st Jan.
>
>   Ok, try this patch. It looks like NTFS accesses auxillary vnodes
>   while flushing other vnodes, so a single vflush scan won't catch
> them
>   all. I don't even know if multiple vflushes will catch them all
>   but lets try it and find out.
>
>   -Matt
>
> diff --git a/sys/vfs/ntfs/ntfs_vfsops.c b/sys/vfs/ntfs/ntfs_vfsops.c
> index d3d80d2..2a8c1a5 100644
> --- a/sys/vfs/ntfs/ntfs_vfsops.c
> +++ b/sys/vfs/ntfs/ntfs_vfsops.c
> @@ -633,6 +633,8 @@ ntfs_unmount(struct mount *mp, int mntflags)
>
> dprintf(("ntfs_unmount: vflushing...\n"));
> error = vflush(mp, 0, flags | SKIPSYSTEM);
> + error = vflush(mp, 0, flags | SKIPSYSTEM);
> + error = vflush(mp, 0, flags | SKIPSYSTEM);
> if (error) {
>   kprintf("ntfs_unmount: vflush failed: %d\n",error);
>   return (error);
> @@ -649,6 +651,8 @@ ntfs_unmount(struct mount *mp, int mntflags)
>
> /* vflush system vnodes */
> error = vflush(mp, 0, flags);
> + error = vflush(mp, 0, flags);
> + error = vflush(mp, 0, flags);
> if (error)
>   kprintf("ntfs_unmount: vflush failed(sysnodes): %d\n",error);
>

Hi,

I tried your patch, but it didn't solve the problem. I'm using a
x86_64 kernel from Jan 30th on xen.

(gdb) bt
#0 Debugger (msg=<value optimized out>)
at /usr/src/sys/platform/pc64/x86_64/db_interface.c:360
#1 0xffffffff8036aa0d in panic (fmt=0xffffffff8064bb8e "unmount:
dangling vnode")
at /usr/src/sys/kern/kern_shutdown.c:742
#2 0xffffffff803d3762 in dounmount (mp=0xffffffff91f64260, flags=0)
at /usr/src/sys/kern/vfs_syscalls.c:783
#3 0xffffffff803d38d0 in sys_unmount (uap=0xffffffffb2f07b48)
at /usr/src/sys/kern/vfs_syscalls.c:618
#4 0xffffffff805c40fa in syscall2 (frame=0xffffffffb2f07bf8)
at /usr/src/sys/platform/pc64/x86_64/trap.c:1189
#5 0xffffffff805bcce3 in Xfast_syscall ()
at /usr/src/sys/platform/pc64/x86_64/exception.S:304
#6 0x00007fffffffa8f in ?? ()
Reply contains invalid hex digit 84

Here's the info I have regarding the ntfs partition I'm testing with:
vm11 /usr/src/sys/vfs/ntfs/
$ df | grep ad2
/dev/ad2   51200   6196   45004
12%   /mnt
vm1 /usr/src/sys/vfs/ntfs/
$ mount | grep ad2
/dev/ad2 on /mnt (ntfs, local)
vm1 /usr/src/sys/vfs/ntfs/

```
$ du  /mnt/
0 /mnt/$Extend
352 /mnt/boot/common
16 /mnt/boot/ficl/i386
16 /mnt/boot/ficl/ia64
81 /mnt/boot/ficl/softwords
16 /mnt/boot/ficl/sparc64
545 /mnt/boot/ficl
112 /mnt/boot/forth
1025 /mnt/boot
3221 /mnt/
```

Tim

#3 - 02/10/2010 09:05 AM - dillon

If someone could get me a small sample image of a NTFS filesystem
that exhibits the problem tracking this down will be easier.

-Matt

#4 - 02/10/2010 05:18 PM - dillon

```
:
:Hi Matt,
:
:Grab from...
:
:Just vnconfig and mount_ntfs it. I've put some dummy data in there.
:
:Cheers,
:Antonio Huete
```

No luck. I couldn't get it to crash.

Hmm. Rumko, this is a bit drastic but here's a patch that
will add serious debugging output to the console. I kept
the old patch too. Please run this and upload a core from
it when it crashes. I may be able to figure out where the
vnodes are being allocated from by looking at the backtrace.

-Matt

```
diff --git a/sys/vfs/ntfs/ntfs_vfsops.c b/sys/vfs/ntfs/ntfs_vfsops.c
index d3d80d2..a57ec3c 100644
--- a/sys/vfs/ntfs/ntfs_vfsops.c
+++ b/sys/vfs/ntfs/ntfs_vfsops.c
@@ -633,12 +633,14 @@ ntfs_unmount(struct mount *mp, int mntflags)

dprintf(("ntfs_unmount: vflushing...\n"));
error = vflush(mp, 0, flags | SKIPSYSTEM);
+ error = vflush(mp, 0, flags | SKIPSYSTEM);
+ error = vflush(mp, 0, flags | SKIPSYSTEM);
if (error) {
kprintf("ntfs_unmount: vflush failed: %d\n",error);
return (error);
}

- /* Check if only system vnodes are rest */
```

```
+ /* Check if only system vnodes are left */
for(i=0;i<NTFS_SYSNODESNUM;i++)
if((ntmp->ntm_sysvn[i]) &&
(ntmp->ntm_sysvn[i]->v_sysref.refcnt > 1)) return (EBUSY);
@@ -649,6 +651,8 @@ ntfs_unmount(struct mount *mp, int mntflags)

/* vflush system vnodes */
error = vflush(mp, 0, flags);
+ error = vflush(mp, 0, flags);
+ error = vflush(mp, 0, flags);
if (error)
kprintf("ntfs_unmount: vflush failed(sysnodes): %d\n",error);

@@ -894,6 +898,8 @@ ntfs_vgetex(struct mount *mp, ino_t ino, u_int32_t attrtype, char *attrname,
}

error = getnewvnode(VT_NTFS, ntmp->ntm_mountp, &vp, VLKTIMEOUT, 0);
+ kprintf("NTFS: getnewvnode %p\n", vp);
+ print_backtrace();
if(error) {
ntfs_frele(fp);
ntfs_ntput(ip);
```

#5 - 02/10/2010 06:41 PM - dillon

All fixed now. Thanks to Tim getting me a nice NTFS image
exhibiting the problem.

It turns out that NTFS used vnode->v_type = VNON for special
internal extent vnodes. VNON vnodes are ignored by vflush()
and thus get left dangling.

-Matt

Files

unnamed	5.67 KB	02/10/2010	tbisson