DragonFlyBSD - Bug #2863

HAMMER synch tid is zero

12/09/2015 11:49 AM - shamaz

Status:	New	**Start date:**	12/09/2015
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Hello. This appears on DragonFlyBSD 4.2 (and maybe newer) if you try the
following with HAMMER filesystem:

1) Create a new vnode device.
```
# dd if=/dev/zero of=test.img bs=1m count=15000
# vnconfig -c vn test.img
# disklabel64  -w -r vn4s0 auto
# disklabel64 -e vn4s0
```
Add a line like "a: *   0   HAMMER"

2) Create a HAMMER filesystem:
```
newfs_hammer -f -L TEST /dev/vn4s0a
```

3) Mount it:
```
mount /dev/vn4s0a /mnt
```

4) Try to sync it:
```
hammer synctid /mnt
```

You will see zero sync tid (0x0000000000000000). If you try to set some
debug-related sysctls, it will get you no info at all when syncing. Also
hammer pfs-status /mnt will show that sync-end-tid does not increment in
time as it should. I am not sure if it happens with any non-root hammer FS,
or just with those based on vn pseudo disks.

History

#1 - 12/09/2015 12:23 PM - tkusumi

the tid you're seeing is what's kept by the filesystem's internal flusher data structure.
it starts off with 0.
unless you have something to sync it's 0.

#2 - 12/09/2015 11:27 PM - shamaz

As far as I know, any amount of subsequent TIDs is at least a non-decrementing sequence (if not strictly incrementing). So if sync-end-tid is not zero (as in my case), may this be, that a subsequent TID is zero? I forgot to mention, that my new FS is not completely new, I created one or two files, a new master PFS and a snapshot (of course with TID=0). Is this by design? I'll check later if the same happens when there is I/O in progress when syncing.

#3 - 12/10/2015 12:20 AM - tkusumi

Literally copy paste what you did and illustrate what your problem is.
(Not by some copy pastes and some sentences followed by another sentences that you forgot to mention)

#4 - 12/10/2015 08:27 AM - shamaz

OK. This is my test case (posted it on pastebin). Just please try to reproduce it to see if it's the same on recent version or master.

http://pastebin.com/Ag36fDxT

If you do not see anything unusual: see carefully on snapshot timestamps and TIDs. Snapshot with transaction ID 0x0000000000000000 is not the earliest. Looks to me, you initialize your "internal flusher data structure" too late, maybe on the first write or something like that. Maybe this is not critical and can be ignored, you decide.

#5 - 12/10/2015 08:39 AM - tkusumi

from a link in comment #4

```
-------------------------------------------------------------------
ressurected# dd if=/dev/zero of=test.img bs=1m count=15000
15000+0 records in
15000+0 records out
15728640000 bytes transferred in 299.641029 secs (52491610 bytes/sec)
ressurected# vnconfig -c vn test.img
vn4
ressurected# disklabel64 -w -r /dev/vn4s0 auto
ressurected# disklabel64 -e /dev/vn4s0
ressurected# newfs_hammer -f -L TEST /dev/vn4s0a
Volume 0 DEVICE /dev/vn4s0a     size  14.65GB
initialize freemap volume 0
initializing the undo map (504 MB)
---------------------------------------------
1 volume total size  14.65GB version 6
boot-area-size:       32.00MB
memory-log-size:      32.00MB
undo-buffer-size:    504.00MB
total-pre-allocated:   0.51GB
fsid:                44b3c7ea-9f56-11e5-a636-11feed02b918

NOTE: Please remember that you may have to manually set up a
cron(8) job to prune and reblock the filesystem regularly.
By default, the system automatically runs 'hammer cleanup'
on a nightly basis.  The periodic.conf(5) variable
'daily_clean_hammer_enable' can be unset to disable this.
Also see 'man hammer' and 'man HAMMER' for more information.

WARNING: HAMMER filesystems less than 50GB are not recommended!
You may have to run 'hammer prune-everything' and 'hammer reblock'
quite often, even if using a nohistory mount.
ressurected# mount /dev/vn4s0a /mnt/
ressurected# cd /mnt/
ressurected# touch it
ressurected# hammer snapq .
./@@0x0000000100008020
ressurected# touch seems
ressurected# hammer snapq .
./@@0x0000000100008080
ressurected# touch to\ be\ working
ressurected# hammer snapq .
./@@0x0000000100008120
ressurected# cd
ressurected# umount /mnt/
ressurected# mount /dev/vn4s0a /mnt/
ressurected# cd /mnt/
ressurected# hammer snapq .
./@@0x0000000000000000
ressurected# touch no\ it\ does\ not
ressurected# hammer snapq .
./@@0x0000000100010220
ressurected# hammer snapls .
Snapshots on .  PFS #0
Transaction ID          Timestamp               Note
0x0000000000000000      2015-12-10 19:17:09 MSK -
0x0000000100008020      2015-12-10 19:16:29 MSK -
0x0000000100008080      2015-12-10 19:16:35 MSK -
0x0000000100008120      2015-12-10 19:16:54 MSK -
0x0000000100010220      2015-12-10 19:17:23 MSK -
```

#6 - 12/10/2015 11:04 AM - shamaz

- File flusher_tid.patch added

OK, I get it now. As you said before hmp->flusher.tid

> starts off with 0.
> unless you have something to sync it's 0.

So if you mount your filesystem and yet has nothing to sync, when you create a new snapshot, its transaction ID will be that of hmp->flusher.tid, that's zero. So if you want it to increase monotonically, you must initialize hmp->flusher.tid earlier. I wrote a patch, please tell me if it is OK.

#7 - 12/11/2015 05:09 AM - tkusumi

haven't look at details, but see if your patch works and your patch is correct on mount update case,
when hmp (as well as flusher which is a part of *hmp) already exists on mount(2).

#8 - 12/11/2015 06:54 AM - shamaz

Again, it's copy-paste time.

```
# mount /dev/vkd1s0a /mnt/
HAMMER(TEST) recovery check seqno=000fbfff
HAMMER(TEST) recovery range 3000000000000000-3000000000000000
HAMMER(TEST) recovery nexto 3000000000000000 endseqno=000fc000
HAMMER(TEST) mounted clean, no recovery needed
# touch /mnt/me
# hammer snapq /mnt/
/mnt/@@0x0000000100008040
# touch /mnt/filename
# hammer snapq /mnt/
/mnt/@@0x00000001000080e0
# mount -u -o ro /mnt
HAMMER read-write -> read-only
# Warning: vfsync skipped 1 dirty bufs in pass2!
mount -u -o rw /mnt/
HAMMER read-only -> read-write
# Warning: vfsync skipped 1 dirty bufs in pass2!
Warning: vfsync skipped 1 dirty bufs in pass2!
hammer snapq /mnt
/mnt@@0x00000001000080e0
# touch /mnt/hammer
# hammer snapq /mnt
/mnt@@0x00000001000081a0
# hammer snapq /mnt
/mnt@@0x00000001000081a0
# umount /mnt
# mount /dev/vkd1s0a /mnt
HAMMER(TEST) recovery check seqno=000fc055
HAMMER(TEST) recovery range 3000000000007d68-3000000000007d68
HAMMER(TEST) recovery nexto 3000000000007d68 endseqno=000fc056
HAMMER(TEST) mounted clean, no recovery needed
# hammer snapq /mnt
/mnt@@0x0000000100008230
# hammer snapls /mnt
Snapshots on /mnt       PFS #0
Transaction ID          Timestamp               Note
0x0000000100008040      2015-12-11 14:28:52 UTC -
0x00000001000080e0      2015-12-11 14:29:22 UTC -
0x00000001000081a0      2015-12-11 14:30:42 UTC -
0x0000000100008230      2015-12-11 14:31:02 UTC -
# umount /mnt
```

As you can see, there is no zero TIDs. The problem is solved, but snapshot with TID 0x0000000100008230 actually should not be there, as there is no writes to filesystem between 0x00000001000081a0 and 0x0000000100008230. I just allocate a new TID for hmp->flusher.tid on mount in order to keep it from being 0. Unfortunately, I cannot set hmp->flusher.tid to a previous value before (re-)mount (which would be the best), because it is not stored on disk. There is two choices: ether just allocate a new tid or write a whole new function to recover hmp->flusher.tid from hmp->next_tid.
Maybe you can show this to dillon to know his opinion.

#9 - 12/11/2015 07:25 AM - tkusumi

Then your patch is just as insane as the existing behavior in terms of tid and syncing, no ?

> The problem is solved

What's your actual problem here which is what i've been keep asking you but you never clearly stated other than saying it's tid=0x0 ?
Do you just not want to see 0x0 in insane order ?
Or does the fact that a snapshot showing 0x0 causing you some filesystem issues other than it saying 0x0 ?

#10 - 12/11/2015 08:15 AM - tkusumi

Also note that the tid value isn't just a number to make hammer snapshots, pfses, etc look nicer via hammer's userspace commands.

It's a core part of mirroring and asof btree, where a larger tid means there was an update to the filesystem's btree that you need to take objects with that tid into account.

It's really a sensitive thing that it needs to be carefully examined.

#11 - 12/12/2015 10:21 PM - dillon

Well, my first thought before looking at the patch was that the initial conditions for hmp->flusher.tid needed to be initialized so a retrieval prior to any actual flush returns a reasonable TID.

The patch appears to do that, but I agree that it probably should not allocate a new TID. We should be able to initialize flusher.tid from the on-disk volume structure. The alloctid code is calculating from hmp->next_tid. hmp->next_tid is initialized from the root volume structure:

```
hmp->next_tid = rootvol->ondisk->vol0_next_tid;
```

So it should be possible to just do an initial assignment of flusher.tid to hmp->next_tid. If that doesn't work (if hmp->next_tid is not yet initialized at that point, but I think is)... then it could be pulled from rootvol->vol0_next_tid using this sequence:

```
volume = hammer_get_root_volume(hmp, &error);
hmp->flusher.tid = volume->ondisk->vol0_next_tid;
hammer_rel_volume(volume, 0);
```

That would prevent the ioctl from improperly returning a TID of 0. A TID of 0 definitely should never be returned by that ioctl.

-Matt

#12 - 12/12/2015 11:24 PM - shamaz

- File hammer_flusher.c.patch added

hmp->next_tid is already initialized when hammer_flusher_create() is called. This is what I get with "hmp->flusher.tid = hmp->next_tid;" :

```
# mount /dev/vkd1s0a /mnt/
HAMMER(TEST) recovery check seqno=000fbfff
HAMMER(TEST) recovery range 3000000000000000-3000000000000000
HAMMER(TEST) recovery nexto 3000000000000000 endseqno=000fc000
HAMMER(TEST) mounted clean, no recovery needed
# cd /mnt/
# touch it
# hammer snapq .
./@@0x0000000100008020
# touch works
# hammer snapq .
./@@0x0000000100008080
# touch so
# hammer snapq .
./@@0x0000000100008120
# cd
# umount /mnt
# mount /dev/vkd1s0a /mnt/
HAMMER(TEST) recovery check seqno=000fc04a
HAMMER(TEST) recovery range 3000000000006920-3000000000006920
HAMMER(TEST) recovery nexto 3000000000006920 endseqno=000fc04b
HAMMER(TEST) mounted clean, no recovery needed
# hammer snapq /mnt/
/mnt/@@0x00000001000081a0
# echo "hi" > /mnt/so
# hammer snapq /mnt/
/mnt/@@0x0000000100008210
# hammer snapq /mnt/
/mnt/@@0x0000000100008210
# hammer snapls /mnt/
Snapshots on /mnt       PFS #0
Transaction ID          Timestamp               Note
0x0000000100008020      2015-12-13 07:12:36 UTC -
0x0000000100008080      2015-12-13 07:12:46 UTC -
0x0000000100008120      2015-12-13 07:12:54 UTC -
0x00000001000081a0      2015-12-13 07:13:11 UTC -
0x0000000100008210      2015-12-13 07:13:41 UTC -
# umount /mnt/
```

No zeros. I send a new patch, called hammer_flusher.c.patch

Files

flusher_tid.patch	492 Bytes	12/10/2015	shamaz
hammer_flusher.c.patch	481 Bytes	12/13/2015	shamaz