FreeNAS - Bug #26650

Correctly report ZFS dataset quota overflows

11/14/2017 07:25 AM - Caleb St. John

Status:	Resolved		
Priority:	Critical		
Assignee:	Alexander Motin		
Category:	OS		
Target version:	11.1-RC3		
Seen in:	11.0-U4	**Needs Merging:**	Yes
Severity:	New	**Needs Automation:**	No
Reason for Closing:		**Support Suite Ticket:**	n/a
Reason for Blocked:		**Hardware Configuration:**	
Needs QA:	No	**ChangeLog Required:**	No
Needs Doc:	Yes		

Description

When writing to a particular ZFS dataset and that dataset being below quota when the files are initially opened for write, but reaching its quota while the clients are actively writing, eventually something in zfs slows to a crawl.

This is trivially reproducible. Running TrueNAS 11.0-U4

1.zfs create cargo/quota-test
2.zfs set compression=off cargo/quota-test
3.zfs set refquota=50m cargo/quota-test
4.cd /mnt/cargo/quota-test
5.dd if=/dev/zero of=file.out bs=1m

If you perform these steps, you will notice that the "dd" command will continue issuing writes to the dataset but will never stop even though the quota has been met. I have procstat -kk -a | fgrep dd and attached the output to this ticket.

NOTE I have tried this on proper freeBSD 11.0, 11.1, and CURRENT and all produce the same results.

Associated revisions

Revision c601305d - 11/22/2017 07:46 AM - avg

make illumos uiocopy use vn_io_fault_uiomove

uiocopy() is currently unused, its purpose is copy data from a uio
without modifying the uio. It was in use before the vn_io_fault support
was added to ZFS, at which point our code diverged from the illumos code
a little bit. Because ZFS is the only (potential) user of the function
we are free to modify it to better suit ZFS needs.

The intention behind this change is to remove the differences introduced
earlier in zfs_write().

While here, re-implement uioskip() using uiomove() with
uio_segflg == UIO_NOCOPY.
The story of uioskip is the same as with uiocopy.

Reviewed by: mav
MFC after: 1 week
Ticket: #26650

(cherry picked from commit 0e4af542392ca3d738baa5191f0e842506a0d5f5)

Revision c601305d - 11/22/2017 07:46 AM - avg

make illumos uiocopy use vn_io_fault_uiomove

uiocopy() is currently unused, its purpose is copy data from a uio
without modifying the uio. It was in use before the vn_io_fault support
was added to ZFS, at which point our code diverged from the illumos code
a little bit. Because ZFS is the only (potential) user of the function
we are free to modify it to better suit ZFS needs.

The intention behind this change is to remove the differences introduced
earlier in zfs_write().

While here, re-implement uioskip() using uiomove() with
uio_segflg == UIO_NOCOPY.
The story of uioskip is the same as with uiocopy.

Reviewed by: mav
MFC after: 1 week
Ticket: #26650

(cherry picked from commit 0e4af542392ca3d738baa5191f0e842506a0d5f5)

Revision 2fc7a5c9 - 11/22/2017 07:47 AM - avg

zfs_write: fix problem with writes appearing to succeed when over quota

The problem happens when the writes have offsets and sizes aligned with
a filesystem's recordsize (maximum block size). In this scenario
dmu_tx_assign() would fail because of being over the quota, but the uio
would already be modified in the code path where we copy data from the
uio into a borrowed ARC buffer. That makes an appearance of a partial
write, so zfs_write() would return success and the uio would be modified
consistently with writing a single block.

That bug can result in a data loss because the writes over the quota
would appear to succeed while the actual data is being discarded.

This commit fixes the bug by ensuring that the uio is not changed until
after all error checks are done. To achieve that the code now uses
uiocopy() + uioskip() as in the original illumos design. We can do that
now that uiocopy() has been updated in r326067 to use
vn_io_fault_uiomove().

Reported by: mav
Analyzed by: mav
Reviewed by: mav
Pointyhat to: avg (myself)
MFC after: 1 week
X-MFC after: r326067
X-Erratum: wanted
Ticket: #26650

(cherry picked from commit 5c5e6af72cd1243dfc44da1dbb8ec8f0bda3cfd6)

Revision 2fc7a5c9 - 11/22/2017 07:47 AM - avg

zfs_write: fix problem with writes appearing to succeed when over quota

The problem happens when the writes have offsets and sizes aligned with
a filesystem's recordsize (maximum block size). In this scenario
dmu_tx_assign() would fail because of being over the quota, but the uio
would already be modified in the code path where we copy data from the
uio into a borrowed ARC buffer. That makes an appearance of a partial
write, so zfs_write() would return success and the uio would be modified
consistently with writing a single block.

That bug can result in a data loss because the writes over the quota
would appear to succeed while the actual data is being discarded.

This commit fixes the bug by ensuring that the uio is not changed until
after all error checks are done. To achieve that the code now uses
uiocopy() + uioskip() as in the original illumos design. We can do that
now that uiocopy() has been updated in r326067 to use
vn_io_fault_uiomove().

Reported by: mav
Analyzed by: mav
Reviewed by: mav
Pointyhat to: avg (myself)
MFC after: 1 week
X-MFC after: r326067
X-Erratum: wanted
Ticket: #26650

(cherry picked from commit 5c5e6af72cd1243dfc44da1dbb8ec8f0bda3cfd6)

Revision 64dd5db0 - 11/23/2017 11:38 AM - avg

make illumos uiocopy use vn_io_fault_uiomove

uiocopy() is currently unused, its purpose is copy data from a uio
without modifying the uio. It was in use before the vn_io_fault support
was added to ZFS, at which point our code diverged from the illumos code
a little bit. Because ZFS is the only (potential) user of the function
we are free to modify it to better suit ZFS needs.

The intention behind this change is to remove the differences introduced
earlier in zfs_write().

While here, re-implement uioskip() using uiomove() with
uio_segflg == UIO_NOCOPY.

The story of uioskip is the same as with uiocopy.

Reviewed by: mav
MFC after: 1 week
Ticket: #26650

(cherry picked from commit 0e4af542392ca3d738baa5191f0e842506a0d5f5)
(cherry picked from commit c601305d70ac23601972729521f2dabc3fa0a8ac)

Revision ac8a2500 - 11/23/2017 11:38 AM - avg

zfs_write: fix problem with writes appearing to succeed when over quota

The problem happens when the writes have offsets and sizes aligned with
a filesystem's recordsize (maximum block size). In this scenario
dmu_tx_assign() would fail because of being over the quota, but the uio
would already be modified in the code path where we copy data from the
uio into a borrowed ARC buffer. That makes an appearance of a partial
write, so zfs_write() would return success and the uio would be modified
consistently with writing a single block.

That bug can result in a data loss because the writes over the quota
would appear to succeed while the actual data is being discarded.

This commit fixes the bug by ensuring that the uio is not changed until
after all error checks are done. To achieve that the code now uses
uiocopy() + uioskip() as in the original illumos design. We can do that
now that uiocopy() has been updated in r326067 to use
vn_io_fault_uiomove().

Reported by: mav
Analyzed by: mav
Reviewed by: mav
Pointyhat to: avg (myself)
MFC after: 1 week
X-MFC after: r326067
X-Erratum: wanted
Ticket: #26650

(cherry picked from commit 5c5e6af72cd1243dfc44da1dbb8ec8f0bda3cfd6)
(cherry picked from commit 2fc7a5c98fe68e41c1a34412beb2e65d8f2261ce)

Revision 64dd5db0 - 11/23/2017 11:38 AM - avg

make illumos uiocopy use vn_io_fault_uiomove

uiocopy() is currently unused, its purpose is copy data from a uio
without modifying the uio. It was in use before the vn_io_fault support
was added to ZFS, at which point our code diverged from the illumos code
a little bit. Because ZFS is the only (potential) user of the function
we are free to modify it to better suit ZFS needs.

The intention behind this change is to remove the differences introduced
earlier in zfs_write().

While here, re-implement uioskip() using uiomove() with
uio_segflg == UIO_NOCOPY.
The story of uioskip is the same as with uiocopy.

Reviewed by: mav
MFC after: 1 week
Ticket: #26650

(cherry picked from commit 0e4af542392ca3d738baa5191f0e842506a0d5f5)
(cherry picked from commit c601305d70ac23601972729521f2dabc3fa0a8ac)

Revision ac8a2500 - 11/23/2017 11:38 AM - avg

zfs_write: fix problem with writes appearing to succeed when over quota

The problem happens when the writes have offsets and sizes aligned with
a filesystem's recordsize (maximum block size). In this scenario
dmu_tx_assign() would fail because of being over the quota, but the uio
would already be modified in the code path where we copy data from the
uio into a borrowed ARC buffer. That makes an appearance of a partial
write, so zfs_write() would return success and the uio would be modified
consistently with writing a single block.

That bug can result in a data loss because the writes over the quota
would appear to succeed while the actual data is being discarded.

This commit fixes the bug by ensuring that the uio is not changed until
after all error checks are done. To achieve that the code now uses
uiocopy() + uioskip() as in the original illumos design. We can do that
now that uiocopy() has been updated in r326067 to use
vn_io_fault_uiomove().

Reported by: mav
Analyzed by: mav
Reviewed by: mav
Pointyhat to: avg (myself)
MFC after: 1 week
X-MFC after: r326067
X-Erratum: wanted
Ticket: #26650

(cherry picked from commit 5c5e6af72cd1243dfc44da1dbb8ec8f0bda3cfd6)

(cherry picked from commit 2fc7a5c98fe68e41c1a34412beb2e65d8f2261ce)

History

#1 - 11/16/2017 07:24 AM - Alexander Motin

- Subject changed from zfs refquota deadlock to ZFS dataset quota overflows not reported correctly

- Status changed from Unscreened to Fix In Progress

- Priority changed from Important to Critical

- Target version set to TrueNAS 11.1-U1

Investigation of the problem brought me to FreeBSD-specific change r298105 by avg@ on 2016-04-16. If quota overflow detected during write, the write will fail, but the error status can be lost, falsely reporting partial completion. As result written data are flowing to nowhere and indefinitely, as fast as CPU can handle the loop.

#2 - 11/18/2017 04:29 AM - Alexander Motin

- Project changed from TrueNAS to FreeNAS

- Category changed from 162 to 129

- Target version changed from TrueNAS 11.1-U1 to 11.1

- Migration Needed deleted (No)

- Support Department Priority deleted (0)

This is not specific to TrueNAS.

#3 - 11/22/2017 07:54 AM - Alexander Motin

- Status changed from Fix In Progress to 19

The patch is committed to FreeBSD head. Started FreeNAS test build for the pull request with Jenkins.

#4 - 11/23/2017 11:42 AM - Alexander Motin

- Status changed from 19 to Ready For Release

#5 - 11/23/2017 12:45 PM - Dru Lavigne

- Subject changed from ZFS dataset quota overflows not reported correctly to Correctly report ZFS dataset quota overflows

#6 - 11/27/2017 07:07 AM - Dru Lavigne

- Target version changed from 11.1 to 11.1-RC2

#7 - 11/29/2017 10:42 AM - Nick Wolff

- Needs QA changed from Yes to No

- QA Status Test Passes FreeNAS added

- QA Status deleted (Not Tested)

Stress tested 20 minutes of looping 50mb writes with zero hangs. Passing test.

#8 - 11/29/2017 01:00 PM - Dru Lavigne

- Target version changed from 11.1-RC2 to 11.1-RC3

#9 - 12/01/2017 11:21 AM - Dru Lavigne

- Status changed from Ready For Release to Resolved

Files

1. start-dd.PNG	5.7 KB	11/14/2017	Caleb St. John
2. no-more-space.PNG	3.94 KB	11/14/2017	Caleb St. John
procstat	1.29 KB	11/14/2017	Caleb St. John