CephFS - Feature #44455

cephfs: add recursive unlink RPC

03/05/2020 10:55 PM - Patrick Donnelly

Status:	New	**% Done:**	0%
Priority:	High		
Assignee:	Venky Shankar		
Category:			
Target version:	v17.0.0		
Source:	Development	**Affected Versions:**	
Tags:		**Component(FS):**	MDS
Backport:		**Labels (FS):**	task(intern), task(medium)
Reviewed:		**Pull request ID:**	

Description

This is a fairly common operation [1] and there's no particular reason we can't support it. The PurgeQueue (I think) is already well architected enough to support this with some modification. In particular, it needs to permit an unlinked directory to have children.

I see two immediate use-cases:

- The pybind/mgr/volumes plugin currently has an asynchronous unlink module that cleans up deleted volumes. It'd be much simpler to just tell the MDS to unlink the directory tree.
- cephfs-shell can provide a command which may be used out-of-band to unlink some subtree (thinking HPC)

Keep in mind we can relax some POSIX consistency requirements: the link counts on all the descendants may not change. Logically, this is just renaming the directory to an internal Trash directory that's slowly purged by the MDS. One moderate challenge that needs addressed is revoking any capabilities to asynchronously create files by clients in the subtree. Likewise, the MDS shouldn't create any files in the unlinked subtree via a create RPC.

[1] For example, HDFS has long had a recursive unlink command.

History

#1 - 03/06/2020 05:26 PM - Patrick Donnelly

- Description updated

- Labels (FS) task(intern), task(medium) added

#2 - 04/06/2020 04:23 PM - Venky Shankar

- Assignee set to Venky Shankar

(self assigning this) will start looking to add this support soon.

#3 - 04/06/2020 10:04 PM - Greg Farnum

Hmm one problem the issue description skips over is that this will need to deal with hard-linked files underneath the directory we're trying to recursively delete. That probably means we have to check the full child tree when queueing it up for deletion, and put everything onto the purge queue individually?

#4 - 04/07/2020 01:27 PM - Venky Shankar

Greg Farnum wrote:

> Hmm one problem the issue description skips over is that this will need to deal with hard-linked files underneath the directory we're trying to recursively delete. That probably means we have to check the full child tree when queueing it up for deletion, and put everything onto the purge queue individually?

We should try to avoid the scan if possible. The purge queue would need to deal with unlinking non-empty directories. For the hardlink case you mention, if for a given directory we can tell that there are no hardlinked files in the subtree, that would be a win (not sure how atm!).

#5 - 04/07/2020 01:53 PM - Greg Farnum

Venky Shankar wrote:

> Greg Farnum wrote:
>
> > Hmm one problem the issue description skips over is that this will need to deal with hard-linked files underneath the directory we're trying to recursively delete. That probably means we have to check the full child tree when queueing it up for deletion, and put everything onto the purge queue individually?
>
> We should try to avoid the scan if possible. The purge queue would need to deal with unlinking non-empty directories.

Well, yes, not having to do a scan would be better. But having the purge queue deal with it probably breaks a whole lot of invariants there. (But maybe not? I haven't looked at that code in much depth!)

> For the hardlink case you mention, if for a given directory we can tell that there are no hardlinked files in the subtree, that would be a win (not sure how atm!).

Yeah we can't tell this right now; it would require propagating some kind of "hardlink children" count up in the rstats but even those are asynchronously-updating and non-blocking so we can't actually rely on them. There's really not a good solution here and I don't think it's something you'll be able to count on.

#6 - 12/19/2020 09:27 PM - Patrick Donnelly

- Target version changed from v16.0.0 to v17.0.0