CephFS - Feature #41302

mds: add ephemeral random and distributed export pins

08/15/2019 06:15 PM - Patrick Donnelly

Status:	Resolved	**% Done:**	0%
Priority:	Urgent		
Assignee:	Sidharth Anupkrishnan		
Category:	Performance/Resource Usage		
Target version:	v16.0.0		
Source:	Development	**Affected Versions:**	
Tags:		**Component(FS):**	MDS
Backport:	octopus	**Labels (FS):**	multimds
Reviewed:		**Pull request ID:**	30592

Description

Background: export pins [1] are an effective way to distribute metadata load for large workloads without the metadata balancer interfering. We have found that you can achieve expected linear metadata throughput scaling (with MDS count). However, teaching users or applications how to effectively apply export pins is not always possible.

So, let's make it possible for the MDS to apply a pinning strategy to subtrees. The pinning does not need to be perfect; even a poor distribution is still generally better than single MDS performance. Caveat: some care must be taken to not create too many subtrees as this could degrade performance.

There are two parts to this ticket:

1. Create a new persistent (directory) inode field "export_ephemeral_distributed". This applies only to the directory and is not hierarchical. Any *direct descendant directory* (i.e. a child directory) has an ephemeral export pin applied to it according to a consistent hash [2] of the child directory inode number. This involves each directory knowing its immediate parent's "ephemeral_export_distributed" value. Any MDS rank can figure out where such a directory should be pinned by knowing the hash (module 360) and the number of ranks (which can be linearly distributed points on the circle).

2. Create a new persistent (directory) inode field "export_ephemeral_random". This is hierarchical like "export_pin". Any CDir (fragment!) loaded into the cache may be ephemerally pinned to a random rank. Like "export_ephemeral_distributed", the random rank is determined by a consistent hash. Notably, if another rank is added or removed then the ephemerally pinned subtrees should be uniformly distributed across the ranks. A directory fragment (CDir) that is pinned in this way will remain pinned for as long as it is in the distributed MDS cache (i.e. some MDS has it in memory).

Tests should verify

- that "export_ephemeral_distributed" is approximately uniform and only applies to the direct descendants
- that "export_ephemeral_random" is heirarchical
- that changing max_mds redistributes approximately 1/N or less of the ephemerally pinned subtrees
- that changing max_mds does not create or remove ephemerally pinned subtrees
- that export_pin overrides an ephemeral pin on a parent directory
- that ephemeral pins override a parent export_pin
- that ephemeral pins can be disabled for a subtree by setting export_ephemeral_random=0.0
- that a standby mds taking up a failed mds's rank gets all the subtrees(consistent hashed) handled by that mds

Performance testing should

- validate that large ephemeral pin changes (due to max_mds changes) do not destabilize the MDS cluster
- identify any performance degradation caused by too many pinned subtrees

[1] https://docs.ceph.com/docs/master/cephfs/multimds/#manually-pinning-directory-trees-to-a-particular-rank
[2] https://en.wikipedia.org/wiki/Consistent_hashing

Related issues:

Related to CephFS - Bug #46302: mds: optimize ephemeral rand pin	**Resolved**
Blocks CephFS - Bug #41541: mgr/volumes: ephemerally pin volumes	**Resolved**
Copied to CephFS - Backport #46201: octopus: mds: add ephemeral random and di...	**Resolved**

History

#1 - 08/15/2019 06:25 PM - Patrick Donnelly

It's worth noting that the only difference between the two options is that export_ephemeral_distributed is not hierarchical and applied only to direct descendant directories 100% of the time whereas export_ephemeral_random is hierarchical and randomly applied (i.e. not 100% of the time). I'm open to names that better reflect how these two options are connected in behavior.

#2 - 08/15/2019 09:51 PM - Patrick Donnelly

- Description updated

#3 - 08/15/2019 09:56 PM - Patrick Donnelly

- Description updated

#4 - 08/16/2019 04:04 PM - Patrick Donnelly

- Description updated

#5 - 08/16/2019 04:04 PM - Patrick Donnelly

Here are some scripts shared by Dan from CERN that can be used to manually test random subtree pinning:
https://github.com/cernceph/ceph-scripts/tree/master/tools/cephfs

#6 - 08/16/2019 06:05 PM - Patrick Donnelly

- Description updated

#7 - 08/16/2019 06:05 PM - Patrick Donnelly

- Description updated

#8 - 08/16/2019 06:25 PM - Sidharth Anupkrishnan

Nice!
I have a doubt regarding how we could use consistent hashing for the 2nd case: "export_ephemeral_random" pinning. Since we are pinning entire subtrees, how should we hash such that every directory/file in the subtree gets pinned to a particular MDS rank(a particular segment in the circle in consistent hashing terms) i.e preserve hierarchical locality?

#9 - 08/16/2019 06:52 PM - Patrick Donnelly

- Assignee set to Sidharth Anupkrishnan

Sidharth, I've discussed this with Doug and we'll be assigning this to you.

Sidharth Anupkrishnan wrote:

> Nice!
> I have a doubt regarding how we could use consistent hashing for the 2nd case: "export_ephemeral_random" pinning. Since we are pinning entire subtrees, how should we hash such that every directory/file in the subtree gets pinned to a particular MDS rank(a particular segment in the circle in consistent hashing terms) i.e preserve hierarchical locality?

We would never want to set export_ephemeral_random=1.0 such that every CDir (directory fragment) is ephemerally pinned. That indeed would compromise any benefits from metadata locality. The percentage would probably be low like 1% or 5%.

Keep in mind that the random chance that a CDir is ephemerally pinned is determined when the CDir is created or loaded into memory. Once it has an ephemeral pin, it remains that way for the duration it is in an MDS cache.

#10 - 08/19/2019 08:27 AM - Sidharth Anupkrishnan

Patrick Donnelly wrote:

> Sidharth, I've discussed this with Doug and we'll be assigning this to you.
>
> Sidharth Anupkrishnan wrote:
>
> > Nice!
> > I have a doubt regarding how we could use consistent hashing for the 2nd case: "export_ephemeral_random" pinning. Since we are pinning entire subtrees, how should we hash such that every directory/file in the subtree gets pinned to a particular MDS rank(a particular segment in the circle in consistent hashing terms) i.e preserve hierarchical locality?
>
> We would never want to set export_ephemeral_random=1.0 such that every CDir (directory fragment) is ephemerally pinned. That indeed would compromise any benefits from metadata locality. The percentage would probably be low like 1% or 5%.
>
> Keep in mind that the random chance that a CDir is ephemerally pinned is determined when the CDir is created or loaded into memory. Once it has an ephemeral pin, it remains that way for the duration it is in an MDS cache.

Sounds good! I'll get on it.

#11 - 08/27/2019 10:38 PM - Patrick Donnelly

- Blocks Bug #41541: mgr/volumes: ephemerally pin volumes added

#12 - 09/26/2019 04:01 PM - Sidharth Anupkrishnan

At a first look, I think there is no need to make ephemeral_export_random_pin an xattr like export_pin because for the Inode that has to be ephemeraly_export pinned, the rank that it should be pinned to can be found out by searching the consistent hash ring in the mds_map and then it can be stored as a stack variable and later the balancer can use it to export. Only added complexity (logarithmic) is that we have to search and check if the inode is already ephemerally pinned EVERY time it is loaded(except of course when ephemeral_export_random = 0.0). Now the reason why we have to do this is for the following case: the MDS fails and a standby takes up the rank. Now as the cache gets loaded, we have to search in the consistent hash ring, if it is already ephemerally export pinned by a rank in order to prevent it from getting ephemerally export pinned to another rank. I think this is better than using it as an xattr in which case we have to rely on journaling and network latencies while updating the ephemeral_export_random_pin.

#13 - 09/26/2019 06:07 PM - Sidharth Anupkrishnan

Sidharth Anupkrishnan wrote:

> At a first look, I think there is no need to make ephemeral_export_random_pin an xattr like export_pin because for the Inode that has to be ephemeraly_export pinned, the rank that it should be pinned to can be found out by searching the consistent hash ring in the mds_map and then it can be stored as a stack variable and later the balancer can use it to export. Only added complexity (logarithmic) is that we have to search and check if the inode is already ephemerally pinned EVERY time it is loaded(except of course when ephemeral_export_random = 0.0). Now the reason why we have to do this is for the following case: the MDS fails and a standby takes up the rank. Now as the cache gets loaded, we have to search in the consistent hash ring, if it is already ephemerally export pinned by a rank in order to prevent it from getting ephemerally export pinned to another rank.
> I think this is better than using it as an xattr in which case we have to rely on journaling and network latencies while updating the ephemeral_export_random_pin.

Going through it again, the running time for searching whether an inode is export ephemerally pinned is not logarithmic but (logN + K) where N - No of ranks and K - average no of inodes pinned to a rank and K can get really huge when the workload grows and having this computational overhead everytime we load it into cache is too much right? xattr doesnt seem like a bad option now. The check can be done in constant time but we'd have to have a journal on update(when it gets pinned and when redistribution happens).

Patrick, Your thoughts?

#14 - 09/27/2019 04:45 AM - Patrick Donnelly

Sidharth Anupkrishnan wrote:

> At a first look, I think there is no need to make ephemeral_export_random_pin an xattr like export_pin because for the Inode that has to be ephemeraly_export pinned, the rank that it should be pinned to can be found out by searching the consistent hash ring in the mds_map and then it can be stored as a stack variable and later the balancer can use it to export.

The two proposed export pin xattrs are booleans. export_ephemeral_distributed applies the policy to the directory's direct descendents. export_ephemeral_random applies the policy to all children. These settings should be persistent. Whether or not a directory is pinned is determined when loading it from RADOS. Similarly, where it is ephemerally pinned can be saved in the CInode struct just like CInode::export_pin. The ephemeral pin is not saved to RADOS but is included when exporting the Inode to another MDS.

#15 - 09/27/2019 04:48 AM - Patrick Donnelly

Sidharth Anupkrishnan wrote:

Sidharth Anupkrishnan wrote:

> At a first look, I think there is no need to make ephemeral_export_random_pin an xattr like export_pin because for the Inode that has to be ephemeraly_export pinned, the rank that it should be pinned to can be found out by searching the consistent hash ring in the mds_map and then it can be stored as a stack variable and later the balancer can use it to export. Only added complexity (logarithmic) is that we have to search and check if the inode is already ephemerally pinned EVERY time it is loaded(except of course when ephemeral_export_random = 0.0). Now the reason why we have to do this is for the following case: the MDS fails and a standby takes up the rank. Now as the cache gets loaded, we have to search in the consistent hash ring, if it is already ephemerally export pinned by a rank in order to prevent it from getting ephemerally export pinned to another rank.
> I think this is better than using it as an xattr in which case we have to rely on journaling and network latencies while updating the ephemeral_export_random_pin.

Going through it again, the running time for searching whether an inode is export ephemerally pinned is not logarithmic but (logN + K) where N - No of ranks and K - average no of inodes pinned to a rank and K can get really huge when the workload grows and having this computational overhead everytime we load it into cache is too much right? xattr doesnt seem like a bad option now. The check can be done in constant time but we'd have to have a journal on update(when it gets pinned and when redistribution happens).

Patrick, Your thoughts?

First you need to probabilistically determine if the inode should be ephemerally pinned, that will probably never be more than 5% in practice. If it is ephemerally pinned, then you need to calculate the hash (O(1)) and determine which rank it applies to, I believe that's O(ranks) (or O(1) since ranks cannot be more than 256) for a naive search.

#16 - 09/27/2019 06:20 AM - Sidharth Anupkrishnan

Patrick Donnelly wrote:

> Sidharth Anupkrishnan wrote:
>
> > At a first look, I think there is no need to make ephemeral_export_random_pin an xattr like export_pin because for the Inode that has to be ephemeraly_export pinned, the rank that it should be pinned to can be found out by searching the consistent hash ring in the mds_map and then it can be stored as a stack variable and later the balancer can use it to export.
>
> The two proposed export pin xattrs are booleans. export_ephemeral_distributed applies the policy to the directory's direct descendents. export_ephemeral_random applies the policy to all children. These settings should be persistent. Whether or not a directory is pinned is determined when loading it from RADOS. Similarly, where it is ephemerally pinned can be saved in the CInode struct just like CInode::export_pin. The ephemeral pin is not saved to RADOS but is included when exporting the Inode to another MDS.

Thanks! That Is correct. I was a bit concerned about the situation where a standby MDS takes up a failed one's role and the cache gets loaded. The state should be maintained and the so we have to check if its already pinned and so which rank. Making these pins a boolean xattr simplifies it a lot and the check boils down to O(logN) or O(1) since N<=256.If this boolean xattr wasn't there, we had to search the rank which it would be pinned to(O(logN)) and then search if that Inode is handled by that rank (O(K)) .

So the export_ephemeral_random = [0 to 1] and export_ephemeral_distributed = [0 to 1], which determines the probability of pinning should be config options that the user must set right?

#17 - 09/30/2019 03:53 AM - Patrick Donnelly

Sidharth Anupkrishnan wrote:

> Patrick Donnelly wrote:
>
> > Sidharth Anupkrishnan wrote:
> >
> > > At a first look, I think there is no need to make ephemeral_export_random_pin an xattr like export_pin because for the Inode that has to be ephemeraly_export pinned, the rank that it should be pinned to can be found out by searching the consistent hash ring in the mds_map and then it can be stored as a stack variable and later the balancer can use it to export.
> >
> > The two proposed export pin xattrs are booleans. export_ephemeral_distributed applies the policy to the directory's direct descendents. export_ephemeral_random applies the policy to all children. These settings should be persistent. Whether or not a directory is pinned is determined when loading it from RADOS. Similarly, where it is ephemerally pinned can be saved in the CInode struct just like CInode::export_pin. The ephemeral pin is not saved to RADOS but is included when exporting the Inode to another MDS.
>
> Thanks! That Is correct. I was a bit concerned about the situation where a standby MDS takes up a failed one's role and the cache gets loaded. The state should be maintained and the so we have to check if its already pinned and so which rank. Making these pins a boolean xattr simplifies it a lot and the check boils down to O(logN) or O(1) since N<=256.

I misspoke. The export_ephemeral_distributed setting is a boolean. The export_ephemeral_random is a float [0.0, 1.0]. Both need saved in RADOS, yes.

If a directory inode is ephemerally pinned, then we just note that in the inode as a boolean flag. It remains pinned until all MDS ranks have dropped it from cache.

> If this boolean xattr wasn't there, we had to search the rank which it would be pinned to(O(logN)) and then search if that Inode is handled by that rank (O(K)) .
>
> So the export_ephemeral_random = [0 to 1] and export_ephemeral_distributed = [0 to 1],

false or true **

> which determines the probability of pinning should be config options that the user must set right?

yes

#18 - 09/30/2019 10:31 AM - Sidharth Anupkrishnan

> If a directory inode is ephemerally pinned, then we just note that in the inode as a boolean flag. It remains pinned until all MDS ranks have dropped it from cache.

But in the case I was talking about(an MDS failing and another standby taking up the rank), won't the cache/inodes be dropped and reloaded by the new MDS from RADOS when requests pile up. In this case isn't it necessary to maintain the state i.e the same subtrees that were pinned here before should be pinned here again. Or do you think, they should be rehashed and re-pinned?

#19 - 09/30/2019 12:38 PM - Patrick Donnelly

Sidharth Anupkrishnan wrote:

> If a directory inode is ephemerally pinned, then we just note that in the inode as a boolean flag. It remains pinned until all MDS ranks have dropped it from cache.

> But in the case I was talking about(an MDS failing and another standby taking up the rank), won't the cache/inodes be dropped and reloaded by the new MDS from RADOS when requests pile up. In this case isn't it necessary to maintain the state i.e the same subtrees that were pinned here before should be pinned here again. Or do you think, they should be rehashed and re-pinned?

That would be a reasonable justification for serializing the ephemeral pin state to the MDS journal (backed by RADOS).

#20 - 01/23/2020 01:16 AM - Patrick Donnelly

- Status changed from New to Fix Under Review

- Backport deleted (nautilus)

#21 - 01/23/2020 01:17 AM - Patrick Donnelly

- Category set to Performance/Resource Usage

- Pull request ID set to 30592

#22 - 02/12/2020 03:04 PM - Sidharth Anupkrishnan

- Description updated

#23 - 05/05/2020 05:15 PM - Patrick Donnelly

- Target version changed from v15.0.0 to v16.0.0

#24 - 06/25/2020 01:26 AM - Patrick Donnelly

- Status changed from Fix Under Review to Pending Backport

- Backport set to octopus

#25 - 06/25/2020 01:26 AM - Patrick Donnelly

- Copied to Backport #46201: octopus: mds: add ephemeral random and distributed export pins added

#26 - 07/02/2020 04:47 PM - Patrick Donnelly

- Status changed from Pending Backport to Resolved

#27 - 07/02/2020 04:47 PM - Patrick Donnelly

- Related to Bug #46302: mds: optimize ephemeral rand pin added