Arvados - Feature #13994

[Keepstore] Fetch blocks from federated clusters

08/08/2018 02:41 PM - Peter Amstutz

Status:	Resolved	**Start date:**	09/12/2018
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2018-09-19 Sprint		

Description

Proxy requests for blocks stored on remote clusters.

- To service a GET request with a remote data hint (and no local signature), skip the usual volume scan and use the usual Keep client logic to fetch the block from the remote cluster indicated by the hint.

See Federated collections.

This issue does not include cache optimizations (e.g., writing a copy of the retrieved data on a local volume).

Subtasks:

Task # 14175: Review 13994-proxy-remote	**Resolved**

Related issues:

Related to Arvados - Feature #13993: [API] Fetch remote-hosted collection by ...	**Resolved**	**09/07/2018**

Associated revisions

Revision 8c30c649 - 09/18/2018 07:42 PM - Tom Clegg

Merge branch '13994-proxy-remote'

refs #13994

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

History

#1 - 08/08/2018 02:41 PM - Peter Amstutz

- Status changed from New to In Progress

#2 - 08/08/2018 02:41 PM - Peter Amstutz

- Status changed from In Progress to New

#3 - 08/08/2018 03:13 PM - Peter Amstutz

- Related to Feature #13993: [API] Fetch remote-hosted collection by UUID added

#4 - 08/08/2018 03:57 PM - Peter Amstutz

- Description updated

#5 - 08/08/2018 04:01 PM - Peter Amstutz

For the purposes of completeness.

Another caching strategy that was discussed was for the API server to sign blocks with both a local signature and remote signature, and keepstore wolud return a locally stored block (if available) or retrieve from remote store.

- The API server can efficiently determine if a collection (by PDH) is known and readable by the user. So, arvados-controller could look at the remote collection record PDH and check if a collection with the same PDH is already available locally. However this means to get the benefit of caching requires explicitly copying the collection, and there's no benefit without a local collection with the same PDH.

- A (new) lookup table in the API server could map blocks to collection PDH. This would make it possible to determine at a block level which blocks are reable by the user. This helps if a remote collection shares some blocks with local collections but are not exactly the same. There are no cache benefits if none of the blocks are known to the API server.

- The mapping of (cluster, block, signature) could be stored in the postgres database instead of locally to keepstore.

#6 - 08/08/2018 05:04 PM - Peter Amstutz

- Description updated

#7 - 08/08/2018 05:06 PM - Tom Morris

- Target version changed from To Be Groomed to Arvados Future Sprints

- Story points set to 3.0

- Release set to 14

#8 - 08/10/2018 03:31 AM - Tom Clegg

- Description updated

#10 - 09/05/2018 03:43 PM - Tom Morris

- Assigned To set to Tom Clegg

- Target version changed from Arvados Future Sprints to 2018-09-19 Sprint

#11 - 09/12/2018 08:14 PM - Tom Clegg

13994-proxy-remote @ 55f6178b9a9a0d165e952eeec9a04d0234299397

As discussed offline, we're acknowledging this won't be especially useful until API starts handing out V2 tokens (#14196).

TODO: document "deploy your /etc/arvados/config.yml on keepstore nodes" in install + upgrading pages.

#12 - 09/13/2018 08:42 PM - Peter Amstutz

```
            remote, ok := cluster.RemoteClusters[remoteID]
```

It occurred to me that this won't work for the case of finding clusters by DNS, as opposed to explicitly configured ones. Shall we introduce cluster.GetRemoteCluster(remoteID) instead of cluster.RemoteClusters[remoteID] ?

#13 - 09/13/2018 08:48 PM - Tom Clegg

- Status changed from New to In Progress

Sure. Fetching by PDH also won't work with clusters that aren't explicitly configured by DNS. I think at this point we should stay focused on the cases where the remotes know about one another in advance.

#14 - 09/14/2018 02:00 AM - Peter Amstutz

Took a while to track this one down (manual testing) -

It turns out that the API server signs blocks using the only last part of the v2 token (specifically, current_api_client_authorization.api_token only contains the secret part), but keepstore checks signatures using the entire v2 token (uuid and all). As a result the remote keepstore returns 403 Forbidden. We need to choose which signing scheme is correct (and make sure it is documented.) Personally I think it would be more consistent if it continued to use the entire token string, which means fixing the API server.

Also, the 403 Forbidden (returned by the remote keepstore) gets turned into a 404 by the local keepstore, which is misleading. For debugging sanity, it should probably return 403.

#15 - 09/14/2018 05:07 PM - Tom Clegg

Yes, signing with the entire string seems simpler -- it means many clients don't need to understand the token format at all. I think this is just a matter of remembering the supplied token in Thread.current (source:services/api/app/middlewares/arvados_api_token.rb) and using that instead of a_c_a.api_token when signing (source:services/api/app/models/collection.rb).

#16 - 09/17/2018 02:36 PM - Tom Clegg

13994-proxy-remote @ 8e7c30852f1cf244ae3c58e93acea705739e8625https://ci.curoverse.com/view/Developer/job/developer-run-tests/896/

- api uses entire v2 token to make/check signatures

#17 - 09/17/2018 03:52 PM - Peter Amstutz

Tom Clegg wrote:

> 13994-proxy-remote @ 8e7c30852f1cf244ae3c58e93acea705739e8625https://ci.curoverse.com/view/Developer/job/developer-run-tests/896/
>
> - api uses entire v2 token to make/check signatures

tools/keep-block-check test is failing which seems like an awful cooincidence

#18 - 09/17/2018 06:39 PM - Tom Clegg

13994-proxy-remote @ d9224c40587a8d3617e7be01f3bb7f801c4b52e4https://ci.curoverse.com/view/Developer/job/developer-run-tests/897/

- revisit async RefreshServiceDiscovery change

#19 - 09/18/2018 01:42 PM - Peter Amstutz

Tom Clegg wrote:

> 13994-proxy-remote @ d9224c40587a8d3617e7be01f3bb7f801c4b52e4https://ci.curoverse.com/view/Developer/job/developer-run-tests/897/
>
> - revisit async RefreshServiceDiscovery change

Tests are passing, except a PhantomJS error.

#20 - 09/18/2018 01:49 PM - Peter Amstutz

In ddc180fa5200d9d8fac59cc5041d7d452b68e6a2

In the previous code, in what cases would the "default" branch of select have been activated, and why is that no longer necessary in the new commit?

If ent.clear is closed, does this panic? (That would prevent wg.Done() from being called).

#21 - 09/18/2018 05:04 PM - Tom Clegg

The default case of a select runs if none of the channels are ready -- in this case, if the channel (size=1) already had a value in it, a refresh was already pending, so there was no need to request another.

The previous change tried to unblock the update function, adding the select and thereby changing the semantics from "request a refresh, and block if needed to ensure previous cache will not be used any more" to just "request a refresh". But the keep-block-check tests were relying on the second part, so this reverts to the blocking version.

If someone closed ent.clear then yes, sending to the closed channel would panic, and the program would exit without calling Done().

#22 - 09/18/2018 05:48 PM - Peter Amstutz

Tom Clegg wrote:

> The default case of a select runs if none of the channels are ready -- in this case, if the channel (size=1) already had a value in it, a refresh was already pending, so there was no need to request another.
>
> The previous change tried to unblock the update function, adding the select and thereby changing the semantics from "request a refresh, and block if needed to ensure previous cache will not be used any more" to just "request a refresh". But the keep-block-check tests were relying on the second part, so this reverts to the blocking version.
>
> If someone closed ent.clear then yes, sending to the closed channel would panic, and the program would exit without calling Done().

Since it is in a goroutine, I don't believe it would cause the program to exit, wouldn't it crash the goroutine, so then the caller would be deadlocked? How about this?

```
        wg.Add(1)
        go func() {
            defer wg.Done()
            ent.clear <- struct{}{}
        }()
```

(Sorry to harp on this, I think I understand what you're trying to do but not exactly why it was needed for this particular branch.)

#23 - 09/18/2018 05:57 PM - Tom Clegg

An unrecovered panic really does crash the program. https://golang.org/ref/spec#Handling_panicshttps://play.golang.org/p/iUzMTzUckzF

#24 - 09/18/2018 06:02 PM - Peter Amstutz

Tom Clegg wrote:

> An unrecovered panic really does crash the program. https://golang.org/ref/spec#Handling_panicshttps://play.golang.org/p/iUzMTzUckzF

Got it.

This LGTM, please merge.

#25 - 09/19/2018 01:29 PM - Tom Clegg

- Status changed from In Progress to Resolved