Arvados - Feature #14259

[SDK] Python collection class uses copy remote block to local keepstore

09/26/2018 03:43 PM - Peter Amstutz

<table>
<thead>
<tr>
<th>Status:</th>
<th>Resolved</th>
<th>Start date:</th>
<th>10/30/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assigned To:</td>
<td>Lucas Di Pentima</td>
<td>% Done:</td>
<td>100%</td>
</tr>
<tr>
<td>Category:</td>
<td></td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Target version:</td>
<td>2018-11-14 Sprint</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description
Python SDK Collection class uses "refresh keep signature" API ([#14199](#14199)) to convert +R token signatures into +A signatures.

Only supports copying block from remote clusters to home cluster. Only supports creating collections on the local cluster.

The Python keep client needs a new method to access the "refresh keep signature" API.

The Collection class will check if ArvadosFile objects have any locators with remote signatures (+R) and uses the refresh keep signature API before creating / updating a collection on the home cluster (Collection.save() operation).

Subtasks:
Task # 14293: Review 14259-pysdk-remote-block-copy

Related issues:
- Related to Arvados - Feature #14199: [keepstore] copy block from remote keeps... Resolved 10/04/2018
- Related to Arvados - Feature #14406: [SDK] Go collection uses copy remote bl... Resolved 11/05/2018

Associated revisions
Revision e59b78c7 - 10/30/2018 06:00 PM - Lucas Di Pentima
Merge branch '14259-pysdk-remote-block-copy'
Refs #14259
Arvados-DCO-1.1-Signed-off-by: Lucas Di Pentima <ldipentima@veritasgenetics.com>

Revision a698257b - 11/09/2018 04:26 PM - Lucas Di Pentima
Merge branch '14259-pysdk-remote-block-copy'
Closes #14259
Arvados-DCO-1.1-Signed-off-by: Lucas Di Pentima <ldipentima@veritasgenetics.com>

History
#1 - 09/26/2018 03:43 PM - Peter Amstutz
- Status changed from New to In Progress

#2 - 09/26/2018 03:43 PM - Peter Amstutz
- Related to Feature #14199: [keepstore] copy block from remote keepstore to local keepstore added

#3 - 09/26/2018 03:44 PM - Peter Amstutz
- Description updated

#4 - 09/26/2018 04:14 PM - Peter Amstutz
- Description updated

#5 - 09/26/2018 06:06 PM - Peter Amstutz
- Description updated

#6 - 09/26/2018 06:07 PM - Peter Amstutz
- Description updated
It just occurred to me that this ticket needs a Go SDK counterpart, because crunch-run in some circumstances will copy block locators directly from input to output, so it also needs to support the translation from +R to +A.

Related to Feature #14406: [SDK] Go collection uses copy remote block to local keepstore added

I'm making some manual testing on the keep client side of this story. First tried using 9tee4 & c97qk to get the block signature "translation" and didn't work (after fixing 9tee4's config and being able to get a remote collection from the pysdk), so I tried running 2 federated arvboxes, with the same result.

With the home cluster token I run curl to get a remote cluster block's signature translated and get this error:

```
$ curl -i http://172.17.0.2:25108/6386dee2604bf10d89895a9896e07c04+1962+R1cnct-568e0ea34658171942c3c67cfb76b908dbe5633595be5c162 -H 'Authorization: Bearer v2/34se5-gj3su-ux1dq4873gx33fc/58f5e80t81htxo04ga76y1xq07pq96b2acnjgjomjg7oth16isk' -H 'X-Keep-Signature: local, 2018-10-26T12:42:52.362-03:00'
HTTP/1.1 502 Bad Gateway
Content-Type: text/plain; charset=utf-8
Vary: X-Keep-Signature
X-Content-Type-Options: nosniff
Date: Fri, 26 Oct 2018 17:37:19 GMT
Content-Length: 30
```

Am I missing something?

Both config.yml files seem to be generated correctly:

```
$ cat ~/.arvbox/cluster1/var/cluster_config.yml
Clusters:
  34se5:
```

NodeProfiles:
  '*':
    arvados-api-server: {Listen: ':8004'}
    arvados-controller: {Listen: ':8003'}
PostgreSQL:
  Connection: {DBName: arvados_development, Host: localhost, Password: 55m929avm21jg5kg7ezhblueex,
              User: arvados, client_encoding: utf8}
  ConnectionPool: 32
RemoteClusters:
  1cnct: {Host: '172.17.0.3:8000', Insecure: true, Proxy: true}
  34se5: {Host: '172.17.0.2:8000', Insecure: true, Proxy: true}

$ cat ~/.arvbox/cluster2/var/cluster_config.yml
Clusters:
  1cnct:
    NodeProfiles:
      '*':
        arvados-api-server: {Listen: ':8004'}
        arvados-controller: {Listen: ':8003'}
    PostgreSQL:
      Connection: {DBName: arvados_development, Host: localhost, Password: eh91xfa6og4ciyoayyso6rfbk,
                  User: arvados, client_encoding: utf8}
      ConnectionPool: 32
RemoteClusters:
  1cnct: {Host: '172.17.0.3:8000', Insecure: true, Proxy: true}
  34se5: {Host: '172.17.0.2:8000', Insecure: true, Proxy: true}

#19 - 10/26/2018 06:21 PM - Lucas Di Pentima
I found the issue, keepstores weren't restarted after editing the override config files.

#20 - 10/30/2018 12:24 PM - Lucas Di Pentima
Updates at 4c669f3d7 - branch 14259-py-sdk-remote-block-copy
Test run: https://ci.curoverse.com/job/developer-run-tests/953/
  ● KeepClient.refresh_signature() makes a HEAD request to copy remote blocks to local keep storage.
  ● Collection class uses refresh_signature() when saving a local collection (or creating a new one) with remote blocks.

#21 - 10/30/2018 05:36 PM - Peter Amstutz
  - Assigned To changed from Lucas Di Pentima to Joshua Randall

Lucas Di Pentima wrote:
Updates at 4c669f3d7 - branch 14259-py-sdk-remote-block-copy
Test run: https://ci.curoverse.com/job/developer-run-tests/953/
  ● KeepClient.refresh_signature() makes a HEAD request to copy remote blocks to local keep storage.
  ● Collection class uses refresh_signature() when saving a local collection (or creating a new one) with remote blocks.

if loop.success():
  if method == "HEAD":
    return blob or True
  else:
    return blob

  ● I think when loop.success() is true blob cannot be None, so I don't know if the special case for HEAD or the 'or' clause are necessary?
  ● I noticed a caching bug. It looks like it saves the result of HEAD operations in the cache. That means if you called head() and then called get() with the same locator, you would get back True or the signed locator, but not the expected blob contents. Please write a test to check this case.
  If you can confirm the bug, a possible fix is to not cache the results of HEAD requests.
  ● There is a request timeout and a "low speed timeout" that will kill the request if the transfer time or rate exceeds a certain threshold. However, HEAD requests don't transfer anything, and block until the data is transferred by the keep server. Please make sure it won't fail if the keep-to-keep transfer is slow.
  ● _copy_remote_blocks() only needs to go through the collection contents once, and can use self.items().
  ● I'm a little concerned about the potential overhead of calling _copy_remote_blocks on every collection save, when in the common case we don't have any remote signatures. I wonder if you could do some benchmarking to get a sense of how expensive this extra check is when saving very large collections.

#22 - 10/30/2018 06:16 PM - Lucas Di Pentima
Merged to master to unblock federation, while continuing to polish the following topics:

Peter Amstutz wrote:
I think when loop.success() is true blob cannot be None, so I don’t know if the special case for HEAD or the 'or' clause are necessary?

Yes, I think you’re right. On the other kind of HEAD requests the returned value was True so there’s no point to return True or True.

I noticed a caching bug. It looks like it saves the result of HEAD operations in the cache. That means if you called head() and then called get() with the same locator, you would get back True or the signed locator, but not the expected blob contents. Please write a test to check this case. If you can confirm the bug, a possible fix is to not cache the results of HEAD requests.

We shouldn’t get the blob contents on any HEAD request, right? I think we just use 'blob' var name because it’s a special case of a GET request. OTOH, I think caching HEAD requests would be beneficial for remote locator translations, right now I avoid requesting a copy operation of the same block more than once by using that remote_blocks dict, but if that gets controlled at keep client level, it may be useful to simplify the Collection class’ code, do you agree?

There is a request timeout and a "low speed timeout" that will kill the request if the transfer time or rate exceeds a certain threshold. However, HEAD requests don't transfer anything, and block until the data is transferred by the keep server. Please make sure it won’t fail if the keep-to-keep transfer is slow.

Ok, I suppose timeouts should still be enforced, but not inexistent transfer rates. I'll check that.

_copy_remote_blocks() only needs to go through the collection contents once, and can use self.items().

Right, I'll fix it so we do only one pass, thanks.

I'm a little concerned about the potential overhead of calling _copy_remote_blocks on every collection save, when in the common case we don't have any remote signatures. I wonder if you could do some benchmarking to get a sense of how expensive this extra check is when saving very large collections.

How about checking on local collections if the manifest_text contains '+R[a-z][5]-' and only then call _copy_remote_blocks()?

#23 - 10/30/2018 06:31 PM - Peter Amstutz
Lucas Di Pentima wrote:

- I noticed a caching bug. It looks like it saves the result of HEAD operations in the cache. That means if you called head() and then called get() with the same locator, you would get back True or the signed locator, but not the expected blob contents. Please write a test to check this case. If you can confirm the bug, a possible fix is to not cache the results of HEAD requests.

We shouldn't get the blob contents on any HEAD request, right? I think we just use 'blob' var name because it's a special case of a GET request. OTOH, I think caching HEAD requests would be beneficial for remote locator translations, right now I avoid requesting a copy operation of the same block more than once by using that remote_blocks dict, but if that gets controlled at keep client level, it may be useful to simplify the Collection class' code, do you agree?

The variable is called blob but it holds whatever was returned by the get() method. If you want to cache HEAD responses, you should add a "signed_locator" field to CacheSlot and have slot.set() take both 'contents' and the response in X-Keep-Locator. The potential bug I want to avoid is when calling head() followed by get() you don't want the get() method returning the locator when it was supposed to be returning the block contents.

- I'm a little concerned about the potential overhead of calling _copy_remote_blocks on every collection save, when in the common case we don't have any remote signatures. I wonder if you could do some benchmarking to get a sense of how expensive this extra check is when saving very large collections.

How about checking on local collections if the manifest_text contains '+R[a-z][5]-' and only then call _copy_remote_blocks()? I think that might be a good idea, but a little bit of benchmarking would let us know for sure. Make sure to use the attribute self._manifest_text, not the manifest_text() method. If the collection wasn't initialized from a manifest the attribute will probably be None, so then you have to call _copy_remote_blocks() to be safe.

Thanks!

#24 - 10/31/2018 03:07 PM - Lucas Di Pentima
- Assigned To changed from Joshua Randall to Lucas Di Pentima
- Target version changed from 2018-10-31 sprint to 2018-11-14 Sprint

#25 - 11/01/2018 03:21 PM - Lucas Di Pentima
Updates at b8addea7d6c84c6462a2743191561470332eeaba13
Addressed suggestions. My idea of matching a remote block locator against col._manifest_text won't work because sometimes that attribute isn't updated while having remote blocks. I'm in the process of gathering performance data, it's taking me a little time to build the testing environment. In the meantime the updates can be reviewed.

#26 - 11/02/2018 03:32 PM - Lucas Di Pentima

After fighting too long with arvbox to set a interestingly large collection, I changed strategies making a script that builds a manifest with lots of empty files, creates the collection with `save_new()` and measure the `save()` call time when already committed, giving me these results:

- Without `_copy_remote_blocks()`: ~2.2e-5 secs (without regards of number of files)
- With `_copy_remote_blocks()`: ~1.2e-5 secs **per file** on the collection (totally unacceptaible).

(tested with 10K, 100K & 1M files)

So this is a clear indication that at least on those committed collections the remote block scanning should be avoided. I'm working on optimizations.

#27 - 11/06/2018 02:43 PM - Lucas Di Pentima

Optimized the use of `_copy_remote_blocks()` at c3b2675


Keeping a flag seems like the right strategy.

- `has_remote_blocks()` should check the `has_remote_blocks` flag
- Both `Collection.copy()` and `Collection.rename()` call `Collection.add()`, so the `has_remote_blocks()` check should go there.
- you should add the file part of `_copy_remote_blocks()` to `ArvadosFile`, then you don't need to do `isinstance()`. In addition you could access the file segments list directly (so you don't incur the overhead of a copy from calling `segments()`) and would be a bit cleaner for updating the segment locator directly.

#28 - 11/08/2018 03:34 PM - Peter Amstutz

Lucas Di Pentima wrote:

```
Optimized the use of `_copy_remote_blocks()` at c3b2675
```

As for the second bulletpoint, I did it that way at first because the `add()` call is done on the target_dir, that may or may not be the current collection. I've now added a recursive method to set `has_remote_blocks` all the way up to the root collection.

#29 - 11/08/2018 08:04 PM - Lucas Di Pentima

Updates at da53a8d90


Addressed above suggestions.

As for the second bulletpoint, I did it that way at first because the `add()` call is done on the target_dir, that may or may not be the current collection. I've now added a recursive method to set `has_remote_blocks` all the way up to the root collection.

#30 - 11/09/2018 04:18 PM - Peter Amstutz

Lucas Di Pentima wrote:

```
Updates at da53a8d90
```

Addressed above suggestions.

As for the second bulletpoint, I did it that way at first because the `add()` call is done on the target_dir, that may or may not be the current collection. I've now added a recursive method to set `has_remote_blocks` all the way up to the root collection.

This LGTM.

#31 - 11/09/2018 04:36 PM - Lucas Di Pentima

- Status changed from In Progress to Resolved

Applied in changeset arvadosia698257bb62e04201acd1d9f96edd094296d5c0.

#32 - 11/13/2018 09:52 PM - Tom Morris

- Release set to 14