Arvados - Bug #9998

Improve performance of API server when keep-balance retrieves collections

09/09/2016 12:21 AM - Joshua Randall

Status:	Resolved	**Start date:**	09/08/2016
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:	API	**Estimated time:**	0.00 hour
Target version:	2017-02-01 sprint		

Description

On our system, keep-balance currently takes 7-8 hours to retrieve all ~5.7M collections from an otherwise unloaded API server. When the system is also busy running crunch jobs, it can take closer to 10 hours. Indeed, the latest keep-balance run reported:
```
2016/09/08 20:53:16 GetCurrentState: took 9h59m9.299166541s
```

In comparison, all collection data can be dumped from the database (on a busy system) in less than five minutes:
```
root@humgen-01-01:~# time echo "COPY (select * from collections) TO STDOUT (format text)" | psql ~~U arvados -W -h localhost arvados_production > /data/tmp/collections.txt~~
~~Password for user arvados:~~
~~real  4m39.610s~~
~~user  0m31.554s~~
~~sys  0m29.065s~~
~~root@humgen-01-01:~# ls -lh /data/tmp/collections.txt~~
~~-rw-r--r--~~ 1 root root 13G Sep  9 00:00 /data/tmp/collections.txt
root@humgen-01-01:~# wc -l /data/tmp/collections.txt
5785865 /data/tmp/collections.txt
```

That's 35949s for the supported API server / keep-balance interaction for an underlying operation that can be completed in 279s. There is clearly some overhead involved in performing each SQL query (currently we are retrieving collections in batches of up to 10000), in serialising the results a json, and performing the http interactions for each batch, but I don't think that it is possible for those alone to cause the API server to add 12785% overhead on top of what the database query takes.

Tom pointed out that keep-balance does not need the authorisation signatures to be added to the manifest block locators, only the block locators themselves - he suggested that eliminating the extra signing operations would alleviate a large portion of the overhead. I think it would be worth trying.

We have 5.7M collections with a total of 6.5M distinct blocks on our system, but the collection manifests contain a total of 161M references to those blocks (i.e. we are making great use of deduplication). It looks like the loops in api/app/models/collection.rb go through each locator in each manifest in turn, signing each reference without any caching (https://github.com/curoverse/arvados/blob/7213d3096cdb5d5e03b559a04f88fcd22a835076/services/api/app/models/collection.rb#L220-L241), so if we can skip that step that by setting a flag in the query asking for unsigned manifest locators, in our case we'd be able to avoid 161M signing operations.

An extra ~8h would be explained by each of the 161M signing operations taking ~0.18ms, which seems within the realm of possibility.

Subtasks:

Task # 10950: Review 9998-no-count-items-available	**Resolved**

Related issues:

Related to Arvados - Story #6830: [API] [keep-balance] Option to return unsig...	**Resolved**	
Related to Arvados - Bug #10517: [CLI] Default return fields should be consis...	**Resolved**	**11/10/2016**
Related to Arvados - Bug #10521: [SDKs] [CLI] "arv collection list" retrieves...	**Duplicate**	**11/11/2016**

Associated revisions

Revision ba300c9e - 01/12/2017 04:41 PM - Tom Clegg

Merge branch '9998-unsigned_manifest'

refs #9998

Revision da0aee75 - 01/24/2017 08:28 PM - Tom Clegg

Merge branch '9998-no-count-items-available'

closes #9998

History

#1 - 10/31/2016 05:35 PM - Joshua Randall

In addition to having an option to not sign the locators for collections, a generic list option to avoid calculating "items_available" should also improve performance of keep-balance when retrieving collections.

#2 - 10/31/2016 09:10 PM - Joshua Randall

- Assigned To set to Joshua Randall

Have started to look at this. I'm not sure how best to pass through the desire to get an unsigned manifest.

I could add a query param, but then there needs to be some plumbing to make that available in the model where the signed_manifest_text method lives. It feels a bit wrong to do it that way given the separation between the model and controller. Additionally, using a query param would mean that existing clients (like the `arv` cli) would not know anything about the parameter and thus testing would be difficult.

An alternative would be to make another virtual column (I've already implemented this way calling the column `unsigned_manifest_text`), and to have it available for selection (i.e. using `arv collection list -s '["unsigned_manifest_text","uuid"]'`) but not included in the default select list. This does work, and I have successfully set the API server to not return unsigned_manifest_text unless it is explicitly requested by the client.

The issue with that is that the python SDK (or perhaps CLI) seems to set the select list explicitly even if the user does not specify `-s` (i.e. the client seems to be checking the discovery document to figure out what attributes are available and selects all of them, meaning it doesn't really honor the API server's default attribute list).

Therefore, if I add my "unsigned_manifest_list" attribute, a call to `arv collection list` without `-s` ends up requesting both manifest_text and unsigned_manifest_text and gets both of them, which is a waste since the second one is extraneous.

Some options to fix this include:
- use a query param instead (in which case I'd appreciate some guidance on how to plumb that through without breaking the model-controller interface) - perhaps the signing operation belongs in the controller instead of embedded in the model?
- fix the CLI so that it does not take it upon itself to set the select list if the user doesn't specify `-s`
- in the collections controller, enforce that only one of manifest_text or unsigned_manifest_text can ever be included in the @select list (if you ask for both, just return manifest_text).

#3 - 10/31/2016 09:23 PM - Joshua Randall

Correction to the above - it appears that the Python SDK does not have the issue with selecting columns even when they aren't specified. The blame there appears to be with the ruby client.

Note the difference:

```
# python
Python 2.7.9 (default, Jun 29 2016, 13:08:31)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import arvados
>>> arvados.api().collections().list().execute()['items'][0]
{u'replication_confirmed': None, u'kind': u'arvados#collection', u'expires_at': None, u'uuid': u'
1xpz6-4zz18-rizx4cwtnrro5fq
', u'modified_at': u'2016-10-31T19:10:21.506322000Z', u'created_at': u'2016-10-31T19:10:21.506943000Z', u'desc
ription': None, u'modified_by_client_uuid': u'1xpz6-ozdt8-ndiamwyuovqsou4', u'owner_uuid': u'
1xpz6-tpzed-9bwbc97aj4p4jzl
', u'properties': {}, u'portable_data_hash': u'83ad3dd91034ed6852f6b03efb28cfcf+54', u'href': u'/collections/1
xpz6-4zz18-rizx4cwtnrro5fq', u'replication_confirmed_at': None, u'etag': u'7gv53a7o4gb2aa8d3g1jjdmlm', u'modif
ied_by_user_uuid': u'1xpz6-tpzed-9bwbc97aj4p4jzl', u'replication_desired': None, u'name': u'test1'}
```

With no select specified, the API server returns a collection without any manifest_text to the Python SDK.

However, the CLI returns:

```
# arv collection list -l 1
{
 "kind":"arvados#collectionList",
 "etag":"",
 "self_link":"",
```

```
 "offset":0,
 "limit":1,
 "items":[
  {
   "href":"/collections/1xpz6-4zz18-rizx4cwtnrro5fq",
   "kind":"arvados#collection",
   "etag":"5kyoz6w20j4khe1awqbh9ad6u",
   "uuid":"1xpz6-4zz18-rizx4cwtnrro5fq",
   "owner_uuid":"1xpz6-tpzed-9bwbc97aj4p4jzl",
   "created_at":"2016-10-31T19:10:21.506943000Z",
   "modified_by_client_uuid":"1xpz6-ozdt8-ndiamwyuovqsou4",
   "modified_by_user_uuid":"1xpz6-tpzed-9bwbc97aj4p4jzl",
   "modified_at":"2016-10-31T19:10:21.506322000Z",
   "name":"test1",
   "description":null,
   "properties":{},
   "portable_data_hash":"83ad3dd91034ed6852f6b03efb28cfcf+54",
   "manifest_text":". 1bb8e23bcda1ae0b2d3cd483dadd341f+9208+A7b91f0a6e2dcce756e3b44359eae40086442792 0@582a2b19
 0:9208:README\n",
   "unsigned_manifest_text":". 1bb8e23bcda1ae0b2d3cd483dadd341f+9208 0:9208:README\n",
   "replication_desired":null,
   "replication_confirmed":null,
   "replication_confirmed_at":null,
   "expires_at":null
  }
 ],
 "items_available":2
}
```


#4 - 11/11/2016 04:10 PM - Tom Clegg

Thanks for working on this.

The "unsigned_manifest_text" approach sounds good to me. I suggested calling it plain_manifest_text on #6830 (which wasn't linked from this issue until now, sorry) but I'm OK with either name.

We now have #10517 for what looks like an "arv" bug, which I'd say shouldn't block this unless adding the new column results in **both** being retrieved, in which case this change would make the existing bug more harmful. (I poked at this briefly but I don't see why it's happening.)

#5 - 11/11/2016 10:54 PM - Tom Morris

Adding the new column does result in both getting returned, as shown in the output in Note 3, nullifying the effect of the plain manifest text, at least for the `arv collection list` command.

#6 - 11/12/2016 02:12 AM - Joshua Randall

https://github.com/curoverse/arvados/pull/51

#7 - 11/12/2016 02:19 AM - Joshua Randall

Because of #10517 with these changes `arv collection list` will return both unsigned_manifest_text and manifest_text (it appears to be using the discovery document to explicitly request all selectable columns). #10517 needs to be fixed before the PR implementing unsigned_manifest_text is merged.

Should I change the name to `plain_manifest_text`? I'm happy to do it if you like but I'm not convinced. Calling it `unsigned_manifest_text` seems to make it more obvious how it differs from`manifest_text` - I imagine there could be some ambiguity over what "plain" manifest_text is, since "plaintext" is a thing and it could arguably be applied to either option.

#8 - 11/12/2016 03:21 PM - Joshua Randall

- % Done changed from 0 to 70

I've now completed implementing a `count` param (default true) in the API server (for the list method against all resource types) which if set to false will not perform the COUNT queries against the database and should offer performance improvements.

For example: `arv collection list --no-count` will return a list of collections without `items_available` included in the output and without the overhead of the database COUNT.

#9 - 11/12/2016 03:21 PM - Joshua Randall

Oops, forgot to include: https://github.com/curoverse/arvados/pull/53

#10 - 01/12/2017 03:40 PM - Tom Clegg

Added some tests, now at 9998-unsigned_manifest 8e6cd14b7884a691a110110b0f366577437c6d9e

https://ci.curoverse.com/job/developer-run-tests/133/

#11 - 01/12/2017 03:40 PM - Tom Clegg

- Status changed from New to In Progress

- Assigned To changed from Joshua Randall to Tom Clegg

#12 - 01/19/2017 08:53 PM - Tom Clegg

9998-no-count-items-available @ e1fd558686c78d6edfd460b7531ec9b559299889https://ci.curoverse.com/job/developer-run-tests/142/

#13 - 01/23/2017 03:40 PM - Tom Clegg

- Target version set to 2017-02-01 sprint

#14 - 01/23/2017 03:42 PM - Tom Clegg

9998-no-count-items-available @ b8de9b3e62e82b806576b237be5f317bf378169fhttps://ci.curoverse.com/job/developer-run-tests/146/

#15 - 01/23/2017 08:33 PM - Tom Clegg

Now at 16fe80b0e93ed8c8416b2dcbc0e2ad49bc850738

#16 - 01/24/2017 03:34 PM - Radhika Chippada

- The count option with potential values of ‘exact’ and ‘none’ seems non-intuitive from API perspective. A boolean value as before seems more desirable. Can we consider renaming it as exclude_count with the default values (false or null) working as before and setting exclusively to true meaning actually omit the counts? If boolean values are not acceptable and strings preferable, may be use ‘include’ or empty string instead of ‘exact’ and ‘omit’ or ‘none’ to mean omit to improve documentation.
- query_test.rb - since we keep talking about potentially removing these “other” object types, wondering if we should add one more set of tests based on specimens. Can we use a different object type instead?
- Do we need documentation update around the filter usage?

#17 - 01/24/2017 06:54 PM - Tom Clegg

Radhika Chippada wrote:

> - The count option with potential values of ‘exact’ and ‘none’ seems non-intuitive from API perspective. A boolean value as before seems more desirable. Can we consider renaming it as exclude_count with the default values (false or null) working as before and setting exclusively to true meaning actually omit the counts? If boolean values are not acceptable and strings preferable, may be use ‘include’ or empty string instead of ‘exact’ and ‘omit’ or ‘none’ to mean omit to improve documentation.

This change happened on the github PR (https://github.com/curoverse/arvados/pull/53). Exact/none leaves room for other choices like "approximate count" or "just tell me whether there are any more". I'd rather not reopen/revert that change, if that's OK...

> - query_test.rb - since we keep talking about potentially removing these “other” object types, wondering if we should add one more set of tests based on specimens. Can we use a different object type instead?

Shrug. Seems to me we still have them unless/until we decide to get rid of them, but easily done.

> - Do we need documentation update around the filter usage?

Yes, updated API docs. Thanks.

8e569c16ba035b131c148441ca5a590fb49811ac

#18 - 01/24/2017 08:14 PM - Radhika Chippada

- “exact” versus “exclude_count”: I do not want to belabor it if there is a future goal of supporting other choices such as "approximate count" etc
- doc/api/methods.html => “This option will always produce a faster response.” => May be “… will produce …” without “always”

LGTM

#19 - 01/24/2017 08:35 PM - Tom Clegg

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:da0aee751d8cb039c9b6b85a03e7d62cb973e3b3.

#20 - 02/17/2017 12:30 PM - Joshua Randall

- File 9998-collection-retrieval-time-comparison.png added

Using the arv cli, I timed the performance of collection retrieval on our system using a query very similar to what keep-balance uses to compare the performance of the new options. I ran 10 trials each of the 4 permutations of the two options for count ("none" and "exact") and manifest ("manifest_text" vs "unsigned_manifest_text") using the following loop:

$ for i in $(seq 1 10); do for count in exact none; do for manifest in manifest_text unsigned_manifest_text; do echo "${i} ${count} ${manifest}" && time arv collection list --count ${count} -l 2500 -s '["uuid","'${manifest}'","modified_at","portable_data_hash","replication_desired"]' --order '["modified_at","uuid"]' > /dev/null; done; done; done

The results are in the attached plot and are summarised here.

count=="exact"
Min. 1st Qu. Median Mean 3rd Qu. Max.
32.2 35.3 42.8 42.4 49.5 55.3

count=="none"
Min. 1st Qu. Median Mean 3rd Qu. Max.
23.1 25.5 32.6 34.3 40.6 57.9

signed manifest:
Min. 1st Qu. Median Mean 3rd Qu. Max.
36.5 40.7 46.4 46.2 50.1 57.9

unsigned manifest:
Min. 1st Qu. Median Mean 3rd Qu. Max.
23.1 25.5 30.4 30.5 35.1 40.1

count="exact" and signed manifest:
Min. 1st Qu. Median Mean 3rd Qu. Max.
45.5 47.6 49.5 49.3 50.4 55.3

count=="none" and signed manifest:
Min. 1st Qu. Median Mean 3rd Qu. Max.
36.5 38.9 40.7 43.0 43.0 57.9

count=="exact" and unsigned manifest:
Min. 1st Qu. Median Mean 3rd Qu. Max.
32.2 33.2 35.2 35.5 37.3 40.1

count=="none" and unsigned manifest:
Min. 1st Qu. Median Mean 3rd Qu. Max.
23.1 25.2 25.5 25.6 26.4 28.7

Based on the median times, the change from count=="exact" to count=="none" is responsible for the queries being ~24% faster, while the change from signed to unsigned manifests makes the query ~34% faster. Overall, going from count=="exact" with signed manifests to count=="none" with unsigned manifests results in a ~48% faster query.

I'm curious about the outliers in the count=="none" with signed manifest timings, in which sometimes those queries took a very long time (at the extreme, longer than the longest query with count=="exact"!). That could possibly be interference during the tests, although that there are two such outliers in that category and none to speak of in any of the other three categories.

In terms of real-world performance, our keep-balance has gone from taking 11h40m to retrieve all collections before the update to taking 6h30m to retrieve them after the update, which is ~44% faster.

Meanwhile, I continue to be able to dump ALL collection data directly from the database (usnig psql) in just over 3m, which is 99.2% (130x) faster than the current time keep-balance takes to query via the API server (6h30m). This is an improvement from before, in which querying the database directly was 99.6% (233x) faster, but there clearly remains a massive amount of overhead related to the API server.

$ time echo "COPY (select * from collections) TO STDOUT (format text)" | psql -U arvados -w -h localhost arvados_production > /dev/null

real 3m5.614s
user 0m39.836s
sys 0m10.406s

To be fair, if also writing it to disk this ends up taking 50% longer (although keep-balance is loading the data into memory so that should not be an issue there):

$ time echo "COPY (select * from collections) TO STDOUT (format text)" | psql -U arvados -w -h localhost arvados_production > /data/tmp/collections.dump

real 5m29.001s
user 0m41.026s
sys 0m36.107s

We now have 8.3m collections:
$ wc -l /data/tmp/collections.dump
8324815 /data/tmp/collections.dump

Which in text format takes 18GB to represent:
$ ls ~~-lh /data/tmp/collections.dump~~
~~-rw-r--r--~~ 1 root root 18G Feb 17 12:11 /data/tmp/collections.dump

I'm guessing the remaining overhead probably comes down to ActiveRecord building and processing 8.3m objects for each of these collections?

I wouldn't advocate actually going around the API server for this (i.e. keep-balance should not talk directly to the database), but I'm starting to think it might be necessary to implement some sort of bulk transfer capability in the API so that tools like keep-balance that need all of something can simply ask for a full export of the data, and the API server could skip any unnecessary filtering or framing of the data and just export it directly. Perhaps a new API micro-service could be written in Go to accomplish that?

Files

9998-collection-retrieval-time-comparison.png	133 KB	02/17/2017	Joshua Randall