Arvados - Bug #13165

arvados-cwl-runner on a large workflow results in long (45s+) api server database queries

03/01/2018 08:28 PM - Joshua Randall

Status:	Resolved	**Start date:**	
Priority:	Normal	**Due date:**	
Assigned To:	Ward Vandewege	**% Done:**	0%
Category:	API	**Estimated time:**	0.00 hour
Target version:	2018-05-09 Sprint		

Description

```
SELECT  collections."uuid", collections."owner_uuid", collections."created_at", collections."modif
ied_by_client_uuid", collections."modified_by_user_uuid", collections."modified_at", collections."
name", collections."description", collections."properties", collections."portable_data_hash", coll
ections."replication_desired", collections."replication_confirmed", collections."replication_confi
rmed_at", collections."delete_at", collections."trash_at", collections."is_trashed" FROM "collecti
ons" WHERE (is_trashed = false) AND (NOT EXISTS(SELECT 1
                  FROM materialized_permission_view
                  WHERE trashed = 1 AND
                  (collections.uuid = target_uuid OR collections.owner_uuid = target_uuid))) AND (
(collections.portable_data_hash = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2230'))  ORDER BY collections.
modified_at desc, collections.uuid LIMIT 1 OFFSET 0
```

EXPLAIN ANALYZE on same:

```
arvados_api_production=> EXPLAIN ANALYZE SELECT  collections."uuid", collections."owner_uuid", col
lections."created_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid",
 collections."modified_at", collections."name", collections."description", collections."properties
", collections."portable_data_hash", collections."replication_desired", collections."replication_c
onfirmed", collections."replication_confirmed_at", collections."delete_at", collections."trash_at"
, collections."is_trashed" FROM "collections" WHERE (is_trashed = false) AND (NOT EXISTS(SELECT 1
                  FROM materialized_permission_view
                  WHERE trashed = 1 AND
                  (collections.uuid = target_uuid OR collections.owner_uuid = target_uuid))) AND (
(collections.portable_data_hash = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2230'))  ORDER BY collections.
modified_at desc, collections.uuid LIMIT 1 OFFSET 0
;
QUERY PLAN|Limit  (cost=0.55..988.35 rows=1 width=833) (actual time=39707.297..39707.297 rows=1 lo
ops=1)

QUERY PLAN|  ->  Nested Loop Anti Join  (cost=0.55..601571.59 rows=609 width=833) (actual time=397
07.297..39707.297 rows=1 loops=1)

QUERY PLAN|        Join Filter: (((collections.uuid)::text = (materialized_permission_view.target_
uuid)::text) OR ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text
))

QUERY PLAN|        ->  Index Scan using index_collections_on_modified_at_uuid on collections  (cos
t=0.55..601548.77 rows=1219 width=833) (actual time=39707.281..39707.281 rows=1 loops=1)

QUERY PLAN|              Filter: ((NOT is_trashed) AND ((portable_data_hash)::text = 'e10c53fb1e1d
bfd0ad0f275dc15f05d4+2230'::text))

QUERY PLAN|              Rows Removed by Filter: 1609314

QUERY PLAN|        ->  Materialize  (cost=0.00..1.49 rows=1 width=82) (actual time=0.013..0.013 ro
ws=0 loops=1)

QUERY PLAN|              ->  Seq Scan on materialized_permission_view  (cost=0.00..1.49 rows=1 wid
th=82) (actual time=0.011..0.011 rows=0 loops=1)

QUERY PLAN|                    Filter: (trashed = 1)
```

```
QUERY PLAN|                     Rows Removed by Filter: 39

QUERY PLAN|Planning time: 0.248 ms

QUERY PLAN|Execution time: 39707.348 ms
```

Associated revisions

Revision d862a83e - 04/27/2018 03:57 PM - Ward Vandewege

Documentation: update PostgreSQL installation instructions

refs #13165

Arvados-DCO-1.1-Signed-off-by: Ward Vandewege <wvandewege@veritasgenetics.com>

History

#1 - 03/01/2018 09:08 PM - Joshua Randall

Output from the referenced query:

```
arvados_api_production=> SELECT  collections."uuid", collections."owner_uuid", collections."created_at", colle
ctions."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modified_at", collections.
"name", collections."description", collections."properties", collections."portable_data_hash", collections."re
plication_desired", collections."replication_confirmed", collections."replication_confirmed_at", collections."
delete_at", collections."trash_at", collections."is_trashed" FROM "collections" WHERE (is_trashed = false) AND
 (NOT EXISTS(SELECT 1
                  FROM materialized_permission_view
                  WHERE trashed = 1 AND
                  (collections.uuid = target_uuid OR collections.owner_uuid = target_uuid))) AND ((collections
.portable_data_hash = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2230'))  ORDER BY collections.modified_at desc, collec
tions.uuid LIMIT 1 OFFSET 0
;
uuid|ncucu-4zz18-tv8fxwndu3rwow6
owner_uuid|ncucu-tpzed-000000000000000
created_at|2018-01-29 12:31:14.283357
modified_by_client_uuid|
modified_by_user_uuid|ncucu-tpzed-000000000000000
modified_at|2018-01-29 12:31:14.283357
name|Container output for request ncucu-xvhdp-gx7383iiczj3r49
description|
properties|{"type":"output","container_request":"ncucu-xvhdp-gx7383iiczj3r49"}
portable_data_hash|e10c53fb1e1dbfd0ad0f275dc15f05d4+2230
replication_desired|
replication_confirmed|
replication_confirmed_at|
delete_at|
trash_at|
is_trashed|f
```

This appears to be a ridiculous situation in which the query planner refuses to use the fast btree indices because of the `LIMIT 1` (changing it to `LIMIT 100` makes it very fast).

I was able to trick it into being fast and giving equivalent results by adding an irrelevant column to the `ORDER BY` clause, resulting in a ~2400x speedup:

```
arvados_api_production=> EXPLAIN ANALYZE SELECT  collections."uuid", collections."owner_uuid", collections."cr
eated_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modified_a
t", collections."name", collections."description", collections."properties", collections."portable_data_hash",
 collections."replication_desired", collections."replication_confirmed", collections."replication_confirmed_at
", collections."delete_at", collections."trash_at", collections."is_trashed" FROM "collections" WHERE (is_tras
hed = false) AND (NOT EXISTS(SELECT 1
                  FROM materialized_permission_view
                  WHERE trashed = 1 AND
                  (collections.uuid = target_uuid OR collections.owner_uuid = target_uuid))) AND ((collections
.portable_data_hash = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2230'))  ORDER BY collections.modified_at desc, collec
tions.uuid, collections.name LIMIT 1 OFFSET 0
;
QUERY PLAN|Limit  (cost=5200.13..5200.13 rows=1 width=833) (actual time=16.655..16.656 rows=1 loops=1)

QUERY PLAN|  ->  Sort  (cost=5200.13..5201.65 rows=610 width=833) (actual time=16.653..16.653 rows=1 loops=1)
```

```
QUERY PLAN|         Sort Key: collections.modified_at DESC, collections.uuid, collections.name

QUERY PLAN|         Sort Method: top-N heapsort  Memory: 25kB

QUERY PLAN|         ->  Nested Loop Anti Join  (cost=62.94..5197.08 rows=610 width=833) (actual time=0.361..16.
309 rows=954 loops=1)

QUERY PLAN|               Join Filter: (((collections.uuid)::text = (materialized_permission_view.target_uuid):
:text) OR ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text))

QUERY PLAN|               ->  Bitmap Heap Scan on collections  (cost=62.94..5174.24 rows=1220 width=833) (actua
l time=0.345..15.947 rows=954 loops=1)

QUERY PLAN|                     Recheck Cond: ((portable_data_hash)::text = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2
230'::text)

QUERY PLAN|                     Filter: (NOT is_trashed)

QUERY PLAN|                     Heap Blocks: exact=828

QUERY PLAN|                     ->  Bitmap Index Scan on index_collections_on_portable_data_hash  (cost=0.00..6
2.63 rows=1361 width=0) (actual time=0.253..0.253 rows=954 loops=1)

QUERY PLAN|                           Index Cond: ((portable_data_hash)::text = 'e10c53fb1e1dbfd0ad0f275dc15f05
d4+2230'::text)

QUERY PLAN|               ->  Materialize  (cost=0.00..1.49 rows=1 width=82) (actual time=0.000..0.000 rows=0 l
oops=954)

QUERY PLAN|                     ->  Seq Scan on materialized_permission_view  (cost=0.00..1.49 rows=1 width=82)
 (actual time=0.013..0.013 rows=0 loops=1)

QUERY PLAN|                           Filter: (trashed = 1)

QUERY PLAN|                           Rows Removed by Filter: 39

QUERY PLAN|Planning time: 0.240 ms

QUERY PLAN|Execution time: 16.699 ms
arvados_api_production=> SELECT  collections."uuid", collections."owner_uuid", collections."created_at", colle
ctions."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modified_at", collections.
"name", collections."description", collections."properties", collections."portable_data_hash", collections."re
plication_desired", collections."replication_confirmed", collections."replication_confirmed_at", collections."
delete_at", collections."trash_at", collections."is_trashed" FROM "collections" WHERE (is_trashed = false) AND
 (NOT EXISTS(SELECT 1
                 FROM materialized_permission_view
                 WHERE trashed = 1 AND
                 (collections.uuid = target_uuid OR collections.owner_uuid = target_uuid))) AND ((collections
.portable_data_hash = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2230'))  ORDER BY collections.modified_at desc, collec
tions.uuid, collections.name LIMIT 1 OFFSET 0
;
uuid|ncucu-4zz18-tv8fxwndu3rwow6
owner_uuid|ncucu-tpzed-000000000000000
created_at|2018-01-29 12:31:14.283357
modified_by_client_uuid|
modified_by_user_uuid|ncucu-tpzed-000000000000000
modified_at|2018-01-29 12:31:14.283357
name|Container output for request ncucu-xvhdp-gx7383iiczj3r49
description|
properties|{"type":"output","container_request":"ncucu-xvhdp-gx7383iiczj3r49"}
portable_data_hash|e10c53fb1e1dbfd0ad0f275dc15f05d4+2230
replication_desired|
replication_confirmed|
replication_confirmed_at|
delete_at|
trash_at|
is_trashed|f
```

Argh, databases.

#2 - 03/01/2018 10:18 PM - Joshua Randall

After running `ANALYZE collections`, the original query is much better off (down to 2s from 40s):

```
arvados_api_production=> ANALYZE collections;
ANALYZE
arvados_api_production=> EXPLAIN ANALYZE SELECT  collections."uuid", collections."owner_uuid", collections."cr
eated_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modified_a
t", collections."name", collections."description", collections."properties", collections."portable_data_hash",
 collections."replication_desired", collections."replication_confirmed", collections."replication_confirmed_at
", collections."delete_at", collections."trash_at", collections."is_trashed" FROM "collections" WHERE (is_tras
hed = false) AND (NOT EXISTS(SELECT 1
                  FROM materialized_permission_view
                  WHERE trashed = 1 AND
                  (collections.uuid = target_uuid OR collections.owner_uuid = target_uuid))) AND ((collections
.portable_data_hash = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2230'))  ORDER BY collections.modified_at desc, collec
tions.uuid LIMIT 1 OFFSET 0
;
QUERY PLAN|Limit  (cost=0.55..1078.10 rows=1 width=833) (actual time=1878.155..1878.156 rows=1 loops=1)

QUERY PLAN|  ->  Nested Loop Anti Join  (cost=0.55..604505.12 rows=561 width=833) (actual time=1878.154..1878.
154 rows=1 loops=1)

QUERY PLAN|        Join Filter: (((collections.uuid)::text = (materialized_permission_view.target_uuid)::text)
 OR ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text))

QUERY PLAN|        ->  Index Scan using index_collections_on_modified_at_uuid on collections  (cost=0.55..6044
88.91 rows=841 width=833) (actual time=1878.137..1878.137 rows=1 loops=1)

QUERY PLAN|              Filter: ((NOT is_trashed) AND ((portable_data_hash)::text = 'e10c53fb1e1dbfd0ad0f275d
c15f05d4+2230'::text))

QUERY PLAN|              Rows Removed by Filter: 1626542

QUERY PLAN|        ->  Materialize  (cost=0.00..1.49 rows=1 width=82) (actual time=0.015..0.015 rows=0 loops=1
)

QUERY PLAN|              ->  Seq Scan on materialized_permission_view  (cost=0.00..1.49 rows=1 width=82) (actu
al time=0.012..0.012 rows=0 loops=1)

QUERY PLAN|                    Filter: (trashed = 1)

QUERY PLAN|                    Rows Removed by Filter: 39

QUERY PLAN|Planning time: 0.787 ms

QUERY PLAN|Execution time: 1878.191 ms
```

However, it is still not as fast as the modified one, which is even better off (down to 3ms from 17ms):

```
arvados_api_production=> EXPLAIN ANALYZE SELECT  collections."uuid", collections."owner_uuid", collections."cr
eated_at", collections."modified_by_client_uuid", collections."modified_by_user_uuid", collections."modified_a
t", collections."name", collections."description", collections."properties", collections."portable_data_hash",
 collections."replication_desired", collections."replication_confirmed", collections."replication_confirmed_at
", collections."delete_at", collections."trash_at", collections."is_trashed" FROM "collections" WHERE (is_tras
hed = false) AND (NOT EXISTS(SELECT 1
                  FROM materialized_permission_view
                  WHERE trashed = 1 AND
                  (collections.uuid = target_uuid OR collections.owner_uuid = target_uuid))) AND ((collections
.portable_data_hash = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2230'))  ORDER BY collections.modified_at desc, collec
tions.uuid, collections.name LIMIT 1 OFFSET 0
;
QUERY PLAN|Limit  (cost=3680.28..3680.28 rows=1 width=833) (actual time=3.030..3.030 rows=1 loops=1)

QUERY PLAN|  ->  Sort  (cost=3680.28..3681.68 rows=561 width=833) (actual time=3.030..3.030 rows=1 loops=1)

QUERY PLAN|        Sort Key: collections.modified_at DESC, collections.uuid, collections.name

QUERY PLAN|        Sort Method: top-N heapsort  Memory: 25kB

QUERY PLAN|        ->  Nested Loop Anti Join  (cost=43.78..3677.47 rows=561 width=833) (actual time=0.322..2.5
85 rows=954 loops=1)

QUERY PLAN|              Join Filter: (((collections.uuid)::text = (materialized_permission_view.target_uuid):
:text) OR ((collections.owner_uuid)::text = (materialized_permission_view.target_uuid)::text))

QUERY PLAN|              ->  Bitmap Heap Scan on collections  (cost=43.78..3661.25 rows=842 width=833) (actual
 time=0.303..2.120 rows=954 loops=1)
```

```
QUERY PLAN|                     Recheck Cond: ((portable_data_hash)::text = 'e10c53fb1e1dbfd0ad0f275dc15f05d4+2
230'::text)

QUERY PLAN|                     Filter: (NOT is_trashed)

QUERY PLAN|                     Heap Blocks: exact=828

QUERY PLAN|                     ->  Bitmap Index Scan on index_collections_on_portable_data_hash  (cost=0.00..4
3.57 rows=952 width=0) (actual time=0.189..0.189 rows=954 loops=1)

QUERY PLAN|                           Index Cond: ((portable_data_hash)::text = 'e10c53fb1e1dbfd0ad0f275dc15f05
d4+2230'::text)

QUERY PLAN|               ->  Materialize  (cost=0.00..1.49 rows=1 width=82) (actual time=0.000..0.000 rows=0 l
oops=954)

QUERY PLAN|                     ->  Seq Scan on materialized_permission_view  (cost=0.00..1.49 rows=1 width=82)
 (actual time=0.015..0.015 rows=0 loops=1)

QUERY PLAN|                           Filter: (trashed = 1)

QUERY PLAN|                           Rows Removed by Filter: 39

QUERY PLAN|Planning time: 0.393 ms

QUERY PLAN|Execution time: 3.099 ms
```

#3 - 03/02/2018 02:45 AM - Tom Morris

- Target version set to To Be Groomed

Without diving into the details, I'll not that up-to-date, accurate, and representative statistics are critical to the query planner's making good decisions (and even then it sometimes can be hard to coerce it to do the right thing).

#4 - 03/09/2018 08:51 AM - Joshua Randall

In my opinion, a resolution to this ticket should include:

- A documentation fix in the install guide that mentions the importance of autovacuum (or regular analyze / vacuum jobs), or possibly that explicitly says that further configuration and tuning of the postgres database is outside the scope of the install guide but that it is critical to achieve good performance from Arvados. The instructions as they currently stand result in a postgres installation with no autovacuum (at least on Ubuntu): http://doc.arvados.org/install/install-postgresql.html
- A fix to queries such as this one using a fix such as the one above (adding extraneous unindexed columns to the order), as that continues to improve performance whether or not good stats are available.

#5 - 04/05/2018 06:03 PM - Peter Amstutz

We've adjusted the query so that the query planner no longer ends up in a degenerate sequential scan, can you check and see if this fixes your problem?

#6 - 04/18/2018 01:57 PM - Peter Amstutz

- Status changed from New to Feedback

#7 - 04/18/2018 01:59 PM - Tom Morris

- Status changed from Feedback to New

- Target version changed from To Be Groomed to 2018-05-09 Sprint

We've fixed (one of) the problematic queries.

Let's make sure that the docs get updated as well.

#8 - 04/25/2018 03:55 PM - Tom Morris

- Assigned To set to Ward Vandewege

#9 - 04/25/2018 04:00 PM - Ward Vandewege

Joshua Randall wrote:

> In my opinion, a resolution to this ticket should include:

- A documentation fix in the install guide that mentions the importance of autovacuum (or regular analyze / vacuum jobs), or possibly that explicitly says that further configuration and tuning of the postgres database is outside the scope of the install guide but that it is critical to achieve good performance from Arvados. The instructions as they currently stand result in a postgres installation with no autovacuum (at least on Ubuntu): http://doc.arvados.org/install/install-postgresql.html

I'm a little surprised by this; autovacuum has been enabled by default since at least PostgreSQL 9.1, cf.

https://www.postgresql.org/docs/9.1/static/runtime-config-autovacuum.html

Can you elaborate a bit?

#10 - 04/27/2018 04:00 PM - Ward Vandewege

- Status changed from New to Feedback

I've made a documentation fix along the lines of what you requested, Josh.

#11 - 05/09/2018 01:57 AM - Ward Vandewege

- Status changed from Feedback to Resolved

#12 - 07/23/2018 06:52 PM - Tom Morris

- Release set to 13