Arvados - Story #8886

[API] Do not block individual API queries on rebuilding the permissions graph

04/05/2016 06:56 PM - Brett Smith

Status:	Resolved	**Start date:**	04/13/2016
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	67%
Category:	API	**Estimated time:**	0.00 hour
Target version:	2016-05-11 sprint		

Description

Functional requirements:

- Do not change the database structure.
- Do not write new Go code. As much as possible, we want to use the existing code, just adapt it to run out of band from the request cycle.
- Updating the permissions cache happens outside the normal request handler cycle. No individual request waits for the permissions cache to be updated before it is handled. Each request just uses the "latest" permissions cache available, at least from the first time that it needs the cache.
  Unless it is unreasonably hard, each request should use the same permissions cache throughout its lifecycle. (In general we want this, and if it's low-hanging fruit, this seems like a good opportunity to make that happen. However, we believe that we do not currently have this behavior today, and fixing this is not *so* pressing that we're willing to hold up the performance improvement on it.)
- This new permissions cache handling is behind a feature flag in application.yml called async_permissions_update. When the flag is falsy, the API server continues to use its current behavior of rebuilding the permissions cache when needed. The default in application.default.yml is false.

Subtasks:

Task # 9090: Write permission rebuild service	**Resolved**
Task # 8978: Review	**Resolved**
Task # 9091: Testing	**Closed**

Related issues:

Related to Arvados - Bug #8787: [API] repositories/get_all_permissions method...	**Resolved**	**03/24/2016**	
Precedes (1 day) Arvados - Story #9186: [API] Test client impact of async_per...	**Closed**	**04/15/2016**	**04/15/2016**
Precedes (1 day) Arvados - Story #9502: [API] Update permissions cache as nee...	**Resolved**	**06/28/2016**	

Associated revisions

Revision 12bfd7a6 - 05/11/2016 03:42 PM - Peter Amstutz

Merge branch '8886-async-permission-update' refs #8886

History

#1 - 04/05/2016 07:01 PM - Brett Smith

- Description updated

- Category set to API

#2 - 04/06/2016 06:48 PM - Brett Smith

- Subject changed from [API] Do not block the handling of individual API queries on rebuilding the permissions graph to [API] Do not block individual API queries on rebuilding the permissions graph

#3 - 04/12/2016 06:43 PM - Brett Smith

- Target version set to Arvados Future Sprints

#4 - 04/19/2016 06:17 PM - Brett Smith

Brett Smith wrote:

> - Does Rails provide an easy way to do this through its API? A sort of atomic "replace cached result" method?

That seems unlikely to meet the requirements. Instead the basic idea is we want to find the cheapest possible way to do something "in the background" (i.e., outside the request cycle) for a Rails server.

- Should we move to eventual consistency?

Yes. At the very least, this seems like a good opportunity to experiment with it and see if clients break to think about whether or not we want to have more eventual consistency throughout the API.

- Should we have a separate background process that listens to the relevant tables and builds a computed table with simple (subject, has_permission_to, object) tuples?

Not if we can help it, at this time. We want the smallest possible implementation that can work for now.

- Should we require a newer version of PostgreSQL that can traverse the permissions graph for us (in C)?
  - This may require adjustments to the way we store permissions in the database generally.

This seems like an implementation detail. A worthy one, but still. First we should get the process out of the request cycle. Once we do that, this seems like a potential optimization.

- It would be nice, but not critical, if this was behind a feature flag. That way we could test it on clusters that we know have large permissions graphs before deploying it more broadly.

We do want it behind a feature flag, because we want to enable this on a single "real" cluster (e.g., not 4xphq or c97qk) without enabling it on other production clusters. We also want to be able to roll it back quickly if it turns out not to work well.

#5 - 04/19/2016 06:44 PM - Brett Smith

- Description updated

#6 - 04/26/2016 04:56 PM - Tom Clegg

Possible approach: Run a separate service that updates the permission graph as needed.

Service pseudocode:

- LISTEN for relevant changes on groups table
- while true:
  - Let new_graph_timestamp = db_current_time
  - Compute new_graph
  - Write new_graph to Rails cache
  - Write new_graph_timestamp to Rails cache
  - Wait for the database to notify about a change to the groups table

Presumably, many test cases expect permissions to be applied right away. We will need to fix them by adding a "wait for all permission updates to finish" mechanism. This seems like a useful API to have in the real server, too. The relevant condition is

- The permission graph has been updated since time T (e.g., the modification time found in an API response), *and*
- The permission service is in the "wait for NOTIFY" state, *and*
- There are no Postgres notifications pending

For purposes of test cases, we could have a synchronous "recompute permission graph" API, with guards similar to the "database reset" API.

#7 - 04/27/2016 07:41 PM - Tom Clegg

- Target version changed from Arvados Future Sprints to 2016-05-11 sprint

#8 - 04/27/2016 08:01 PM - Peter Amstutz

- Assigned To set to Peter Amstutz

#9 - 04/27/2016 08:02 PM - Peter Amstutz

- Story points set to 2.0

#10 - 05/04/2016 07:30 PM - Tom Clegg

Should remove the unused copy-and-paste stuff like dispatch_argv and "...the logs table, then push the notification onto the EventMachine channel".

If we get 10x "notify" in a row, do we recalculate the permissions 10x? Postgres says "notifications from different transactions will never get folded into one notification" which seems to mean we need to dedup them ourselves, otherwise it seems like we can easily end up with a backlog of useless recalculations based on notifications that are already older than our cache. The new_graph_timestamp idea from note-6 might help with that.

In permission-updater, the LISTEN statement should be issued before the initial calculation so we don't miss notifications.

If the rails cache expires (or just hasn't been populated yet) is there some reasonably easy way for group_permissions to notify the updater (in case it's "cache expired") and then wait for the updater to finish? This should avoid stuff like "server just came up, and now has lots of threads all trying to recompute the permission graph for themselves, which is even slower because there are so many threads doing it"...

#11 - 05/10/2016 02:26 PM - Peter Amstutz

Tom Clegg wrote:

> Should remove the unused copy-and-paste stuff like dispatch_argv and "...the logs table, then push the notification onto the EventMachine channel".

Yep, fixed.

> If we get 10x "notify" in a row, do we recalculate the permissions 10x? Postgres says "notifications from different transactions will never get folded into one notification" which seems to mean we need to dedup them ourselves, otherwise it seems like we can easily end up with a backlog of useless recalculations based on notifications that are already older than our cache. The new_graph_timestamp idea from note-6 might help with that.

Added last_updated_permissions which would help with that.

> In permission-updater, the LISTEN statement should be issued before the initial calculation so we don't miss notifications.

Done.

> If the rails cache expires (or just hasn't been populated yet) is there some reasonably easy way for group_permissions to notify the updater (in case it's "cache expired") and then wait for the updater to finish? This should avoid stuff like "server just came up, and now has lots of threads all trying to recompute the permission graph for themselves, which is even slower because there are so many threads doing it"...

Now it polls the cache until it gets a non-nil result. It's a bit of a hack but it seems to work.

#12 - 05/11/2016 01:28 PM - Tom Clegg

at e711e48...

The comments in user.rb should be updated: the explanation of how the graph is built should be on calculate_group_permissions, not the method that just returns a cached result.

All test suites are failing when test_helper calls invalidate_permissions_cache.

Presumably the group_permissions method still needs to call calculate_group_permissions if async_permissions_update is false, or something... otherwise the default configuration will just hang?

#13 - 05/11/2016 03:41 PM - Tom Clegg

9e17d8d... LGTM thanks

#14 - 05/11/2016 05:40 PM - Peter Amstutz

- Status changed from New to In Progress

#15 - 05/11/2016 06:12 PM - Brett Smith

- Status changed from In Progress to Resolved

Now that the code specified by the story is done and merged, #9186 is the follow-up story to test it and see how viable it is in a production-like deployment.