Arvados - Story #9502

[API] Update permissions cache as needed after select writes

06/28/2016 05:00 PM - Brett Smith

Status:	Resolved	**Start date:**	06/28/2016
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Today the API server holds its expanded permissions graph in the Rails cache. Any request that needs the permissions graph when it isn't in the cache will generate it. Writes that affect the permissions graph invalidate this change.

Move to a model where we generate the graph on startup, and after each write that affects the graph, before returning a result for that write request. Functional requirements:

- The API server always has an expanded permissions graph cached. It generates one at startup. When it handles a write request that changes the permissions graph, before it returns the result of the request, it generates a new permissions graph that atomically replaces the old one.
- When the async_permissions_update setting is true, incoming requests use whatever copy of the permissions graph is currently complete, without waiting for updates. When the setting is false, if the request needs to use the permissions graph while an update is being prepared, the request waits for that update to finish, then uses the new graph. This wait happens at most once per request.
- Only one update should run at a time, and each update should unblock any write requests that made their underlying database update before the graph rebuild started. For illustration, the implementation should allow this timeline of events:
  1. API server receives write request A.
  2. Write request A updates the database.
  3. API server begins updating the permissions graph.
  4. Write request B comes in and updates the database.
  5. Write request C comes in and updates the database.
  6. Permissions graph update finishes.
  7. Send the result for write request A.
  8. API server begins updating the permissions graph.
  9. Permissions graph update finishes.
  10. Send the result for write requests B and C.

The primary motivation for this branch is to make #9186 more practical. Right now we know it will break clients that make a permissions change, then make an API request that relies on that permissions change to be effective. The idea here is that blocking writes on permission updates should make it possible for those clients to continue working unmodified, without blocking large numbers of readers on the graph update.

We believe this change will also provide performance improvements when async permissions updates are not enabled, just by avoiding redundant graph rebuilds, but that's less of a priority.

Implementation plan for the writer:

1. Commit all changes to the database. Doesn't matter whether it's one or more transactions.
2. Get the current time.
3. Subscribe to notifications on the graph updater timestamp.
4. In a blocking loop, keep reading notifications from the subscription until the timestamp is >= the time we remembered in step 2.
5. Return the API response.

Implementation plan for the updater:

1. Subscribe to relevant changes to the API server database. Remember the times of events that come in from the subscription.
2. When a new event comes in, if its timestamp is older than the most recent mtime of the permissions graph, discard it.
3. Otherwise, in a single database transaction, get the current time, generate the expanded permissions graph, write it to the graph table, and write the remembered time as the new graph's mtime.

Related issues:

Related to Arvados - Story #9186: [API] Test client impact of async_permissio...	**Closed**	**04/15/2016**	**04/15/2016**
Related to Arvados Epics - Story #9053: Port API server to Go	**New**	**04/01/2021**	**09/30/2021**

Follows (1 day) Arvados - Story #8886: [API] Do not block individual API quer...	**Resolved**	**04/13/2016**	

History

#1 - 06/28/2016 05:13 PM - Brett Smith

- Description updated

- Due date deleted (04/15/2016)

- Start date changed from 04/15/2016 to 06/28/2016

#2 - 07/06/2016 06:59 PM - Brett Smith

- Description updated

#3 - 07/19/2017 07:30 PM - Tom Clegg

- Status changed from New to Resolved

Obsolete/unnecessary now that we use a recursive query for permission checks.