Foreman - Bug #18561

Slow exec of delete_removed_facts on MySQL 5.6

02/19/2017 01:18 PM - Yakir Gibraltar

Status: Closed
Priority: Normal
Assignee: Lukas Zapletal
Category: Importers
Target version: 1.15.3
Difficulty: Pull request: https://github.com/theforeman/foreman/pull/4431
Triaged: Fixed in Releases:
Bugzilla link: Found in Releases: 1.14.1

Description
Hi,
I'm using MySQL 5.6 and 2657 hosts, after upgrade foreman to 1.14.1 i see bad performance from DB side on delete_removed_facts in app/models/fact_value.rb one delete take 10 sec instead 1 sec (maximum).
For example:
2017-02-19 15:16:25 04030062 [audit] [I] [461790d9.d462ad46.com] deleted 0 (9844.4ms)
2017-02-19 15:16:25 04030062 [audit] [I] [461790d9.d462ad46.com] updated 10 (126.8ms)
2017-02-19 15:16:25 04030062 [audit] [I] [461790d9.d462ad46.com] added 0 (2.1ms)

Base on https://github.com/theforeman/foreman/commit/46e1ea9f704f41afbd0373e4b8fc54313ac6b397, i reverted:
@ def delete_removed_facts
ActiveSupport::Notifications.instrument "fact_importer_deleted.foreman", :host_id => host.id, :host_name => host.name, :facts => facts, :deleted => [] do [payload] # deletes all facts using a single SQL query (with inner query)
payload[:count] = counters[:deleted] = FactValue.joins(:fact_name).where(:host => host, 'fact_names.type' => fact_name_class).where.not('fact_names.name' => facts.keys).delete_all
end
end

To:
@def delete_removed_facts
to_delete = host.fact_values.eager_load(:fact_name).where("fact_names.type = '#{fact_name_class}' AND fact_names.name NOT IN (?)", facts.keys) # N+1 DELETE SQL, but this would allow us to use callbacks (e.g. auditing) when deleting.
deleted = to_delete.destroy_all
counters[:deleted] = deleted.size
@db_facts = nil
logger.debug("Merging facts for '#{host}': deleted #{counters[:deleted]} facts")
end

After revert this change i see high performance from DB side.
Any option that someone else will test it on MySQL or improve delete_removed_facts?

Thanks!
Yakir.

Related issues:
Related to Foreman - Bug #9016: Fact import code consumes lot of memory Closed 01/18/2015

Associated revisions
Revision b48c2a7f - 04/05/2017 07:23 AM - Lukas Zapletal
Fixes #18561 - faster fact deletion on MySQL

Revision 3155e6b7 - 07/21/2017 01:18 PM - Lukas Zapletal
Fixes #18561 - faster fact deletion on MySQL
History

#1 - 02/19/2017 04:49 PM - Anonymous
- Category set to Importers

#2 - 02/19/2017 04:50 PM - Anonymous
- Related to Bug #9016: Fact import code consumes lot of memory added

#3 - 02/20/2017 03:11 AM - Dominic Cleal
- Legacy Backlogs Release (now unused) set to 221

#4 - 02/20/2017 08:50 AM - Dominic Cleal
- Tracker changed from Refactor to Bug

#5 - 03/02/2017 04:15 AM - Dominic Cleal
- Legacy Backlogs Release (now unused) changed from 221 to 227

#6 - 03/31/2017 08:25 AM - Lukas Zapletal
I don't have MySQL instance handy, but what SQL does it generate? It should be pretty fast as we limit the values via 'host_id' so this should bring number of rows down pretty quick. MySQL optimizer could maybe do it in incorrect order, there are three where clauses.

#7 - 03/31/2017 09:56 AM - Yakir Gibraltar
Lukas Zapletal wrote:

The old version executed (fast transaction):
```
SELECT `fact_values`.id AS t0_r0,
    `fact_values`.value AS t0_r1,
    `fact_values`.fact_name_id AS t0_r2,
    `fact_values`.host_id AS t0_r3,
    `fact_values`.updated_at AS t0_r4,
    `fact_values`.created_at AS t0_r5,
    `fact_names`.id AS t1_r0,
    `fact_names`.name AS t1_r1,
    `fact_values`.updated_at AS t1_r2,
    `fact_names`.created_at AS t1_r3,
    `fact_names`.compose AS t1_r4,
    `fact_names`.short_name AS t1_r5,
    `fact_names`.type AS t1_r6,
    `fact_names`.ancestry AS t1_r7
FROM   `fact_values`
LEFT OUTER JOIN `fact_names`
ON `fact_names`.id = `fact_values`.fact_name_id
WHERE  `fact_values`.host_id = 2309436
AND ( fact_names.type = 'PuppetFactName'
```

03/26/2022
DELETE FROM `fact_values` WHERE `fact_values`.`id` = 410225436

The new version executing (long transaction):

```
DELETE FROM `fact_values` WHERE  `fact_values`.`id` = (SELECT `fact_values`.`id` AS t0_r0,
          `fact_values`.`value` AS t0_r1,
          `fact_values`.`fact_name_id` AS t0_r2,
          `fact_values`.`host_id` AS t0_r3,
          `fact_values`.`updated_at` AS t0_r4,
          `fact_values`.`created_at` AS t0_r5,
          `fact_names`.`id` AS t1_r0,
          `fact_names`.`name` AS t1_r1,
          `fact_names`.`created_at` AS t1_r2,
          `fact_names`.`compose` AS t1_r3,
          `fact_names`.`short_name` AS t1_r4,
          `fact_names`.`type` AS t1_r5,
          `fact_names`.`ancestry` AS t1_r6
FROM   `fact_values`
LEFT OUTER JOIN `fact_names`
ON `fact_names`.`id` = `fact_values`.`fact_name_id`
WHERE
  `fact_values`. `host_id` = 2309436
AND ( fact_names.type = 'PuppetFactName'
```

I tried to collect stats and analyze all foreman schema and it not solved the problem.

I saw few ideas to improve the delete, instead nested subquery:

- http://stackoverflow.com/a/17100006
- http://stackoverflow.com/a/7361345
- http://dba.stackexchange.com/a/71233
I understand, problem is that we are using Rails AR DSL which does not allow to do DELETE FROM and JOINS together. We traded inner query for much more memory effective code.

I think the best way is to file a PR that will do the delete with join but in SQL so no inner query is used. You cannot use delete_all(where) for that, you need to render the SQL manually with to_sql.

Hmmm looks like this is not possible in Rails, the code would be very ugly. See my patch, can you try it?

Thanks Lukas Zapletal, i'll try tomorrow to commit https://github.com/theforeman/foreman/pull/4431 on our foreman.
Lukas Zapletal, i tested your PR and it's fixed the issue.

- Status changed from Ready For Testing to Closed
- % Done changed from 0 to 100

Applied in changeset b48c2a7fb30d4857d0adc39208967ed9d817fc55.

- Legacy Backlogs Release (now unused) set to 241

- Legacy Backlogs Release (now unused) changed from 241 to 276