Redmine - Defect #24786

Heavy performance problem when editing issues (big number of users & projects)

2017-01-09 17:29 - Vincent C.

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Performance	**Estimated time:**	0.00 hour
Target version:			
Resolution:	Invalid	**Affected version:**	3.2.0

Description

Hello !

I'm part of a team maintaining a "big" Redmine instance, and we experience heavy performance impact on some actions, especially when sending an email after an issue update.

Here are some info about our infrastructure :

```
- 8 webservers (2 are dedicated to webservices), all servers are identical :

    * Hardware :
        - 4 vCPU
        - 12 Gb RAM
    * Software :
        - Nginx 1.8.1
        - Ruby 2.0.0
        - Unicorn 4.8.3
            => 4 workers on each server
        - Unicorn Worker Killer 0.4.3
            => Used to kill workers when RAM usage skyrockets
        - Redmine 3.2.0

- ~1000 projects ; ~1000 active users
- There are also ~10 groups which contains a total of ~350 users, and who are members of almost ev
ery project
```

Our problem occurs when an issue is updated/created, which triggers an email sending.
Because of the large number of users/projects, Redmine was taking up to 2 minutes to send an email, which left the user hanging for 2 minutes when saving an issue.
To avoid that, my predecessor edited the Redmine mailing process so that emails are sent in a new thread, which allow Redmine to send the response back immediately to the user, who doesn't have to wait for 2 minutes.
This didn't fix the performance problem, it just hid it to the users.

This week, I decided to take a deeper look at the problem, so I turned debug logs on, in order to find what is taking so long when sending an email.
(the debug log is attached)

Here is what happens in the debug log :

- **lines 1-56** : everything is ok
- **lines 57-439** : All members of the current project are loaded, individually
=> This results in ~400 individual requests (responses come from cache most of the time)
- **Lines 440-542** : everything is ok : Member, MemberRoles & Roles are loaded for each "needed" user
- **Lines 543-640** (activate word wrapping to instantly notice the problem) :
=> For each "needed" user, **all their projects are loaded**
=> The request loading their MemberRoles includes* thousands of MemberRoles ids*
- **Lines 641-682** : everything is ok

This is not really a problem most of the time, but when bulk-editing, the server just can't handle the requests, and we must add

crappy *sleep()* instructions in our scripts ... (This is avoided by running these scripts at night.)
But we can't avoid users bulk-editing issues themselves. And when they do, the server just 'explodes', and the server starts throwing HTTP 500 errors to all users connected to the server.

This is the performance graph when bulk-editing 5 issues on a freshly restarted server (so, nothing is in cache yet) :
!!

If I close 10 issues at the same time, I start getting HTTP 500, and the server is "dead" for the next 5-10 minutes

I tried looking into the code, to find what was causing these horrible delays (and huge amounts of SQL requests), but my knowledge of Rails is not good enough ...
I know that reducing the number of people who are in almost every project would help greatly, but it is not possible for us ATM.
Any help appreciated !

Also, is there any option that I'm not aware of, which can be used to avoid sending completely ?
We use *config.action_mailer.perform_deliveries = false* on non-production environments ; it stops the server from sending the email, but it doesn't avoid the mess explained above
(Redmine still gets all projects/users, loads lot of useless data, just to not send the email in the end)

History

#1 - 2017-01-30 15:49 - Toshi MARUYAMA

- Status changed from New to Needs feedback

You can use :async_smtp. Ref: EmailConfiguration

#2 - 2017-03-17 11:39 - Vincent C.

(*Solved*)

The problem was caused by one of our plugins, which was iterating through each user.memberships and then through each membership.roles when sending a mail, instead of using an SQL join :

```
user.memberships.any? do |membership|
  membership.roles.any? do |role|
    valid_roles.include? role
  end
end
```

I replaced it with a proper ActiveRecord query, and the mail sending is now performing infinitely faster :

```
user.membership(project_id).roles.map(&:id)
```

Toshi MARUYAMA wrote:

> You can use :async_smtp. Ref: EmailConfiguration

Thanks ! Didn't know that Redmine had built-in mail queue

#3 - 2017-03-18 04:03 - Go MAEDA

- Status changed from Needs feedback to Closed

- Resolution set to Invalid

Thank you for reporting the detail.
I am closing this issue.

Files

mail_threading.patch	1.09 KB	2017-01-09	Vincent C.
perf.png	7.84 KB	2017-01-09	Vincent C.
debug.log	337 KB	2017-01-09	Vincent C.