Redmine - Defect #9472

The git scm module causes an excess amount of DB traffic.

2011-10-26 08:03 - John Kubiatowicz

Status:	Closed	**Start date:**	2011-10-26
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	SCM	**Estimated time:**	0.00 hour
Target version:	1.4.0		
Resolution:	Fixed	**Affected version:**	

Description

The fetch_changesets routine from the git scm module (app/model/repository/git.rb) executes a self.save after processing every change. This behavior causes the loading of large repositories to take a long time.

More importantly, this behavior cases problems with plugins that set up observers for the XXX_save events on the repository model (case in point: the redmine_git_hosting plugin). The result is that these callbacks are called far too frequently and end up taking so long that requests to view large repositories can time out and fail under the apache fcgi module.

One fix would be to wait until the very end of the routine to execute self.save. I've attached a proposed patch. With this patch, loading of repositories works much faster and do not timeout.

About your application's environment
Ruby version 1.8.7 (x86_64-linux)
RubyGems version 1.8.11
Rack version 1.1
Rails version 2.3.14
Active Record version 2.3.14
Active Resource version 2.3.14
Action Mailer version 2.3.14
Active Support version 2.3.14
Application root /source/quantum/redmine
Environment production
Database adapter mysql
Database schema version 20110928183125

About your Redmine plugins
Redmine Git Hosting Plugin 0.4.2
Redmine OpenID Selector plugin 0.0.1

Additionally, the redmine code was updated from the trunk this morning (svn ID 7654). Running under apache with the mod_fcgi.

Related issues:

Related to Redmine - Defect #7146: Git adapter lost commits before 7 days fro...	**Closed**	**2010-12-21**
Related to Redmine - Defect #8857: Git: Too long in fetching repositories aft...	**Closed**	**2011-07-20**
Related to Redmine - Defect #6013: git tab,browsing, very slow -- even after ...	**Closed**	**2010-08-02**

Associated revisions

Revision 7658 - 2011-10-27 02:35 - Toshi MARUYAMA

scm: git: recovery and improve comments of fetching from 1.1 about harmful influence that git does not have the revision number (#9472)

Revision 7660 - 2011-10-27 03:31 - Toshi MARUYAMA

scm: git: fix typo of comments about fetching revisions (#9472)

Revision 8840 - 2012-02-11 06:42 - Toshi MARUYAMA

scm: git: reduce saving heads times in fetching revisions (#8857, #9472)

Revision 9141 - 2012-03-07 06:57 - Toshi MARUYAMA

scm: git: backout r8840 (#8857, #9472)

reduce saving heads times in fetching revisions.

Revision 9143 - 2012-03-07 06:57 - Toshi MARUYAMA

scm: git: reduce saving heads times in fetching revisions (#8857, #9472)

Revision 9144 - 2012-03-07 08:56 - Toshi MARUYAMA

scm: git: performance improvements in fetching revisions (#8857, #9472)

Parse a revision for a given branch,
just if we haven't parsed it for any branches before.
Moved the db check to for existing revisions into a grouped search.
Search for many revisions at once: this reduces db load.
Revisions are grouped into sets of 100.
This is to improve memory consumption.
There will be just one query instead of each 100.
The above two methods significantly increase parsing speed.
Test case was a git repo with 6000+ commits on a master branch,
and several other branches originating for master.
Speed improved from 1.4h to 18min.

Contributed by Gergely Fábián.

History

#1 - 2011-10-26 10:36 - Toshi MARUYAMA

- Assignee set to Toshi MARUYAMA

#2 - 2011-10-26 11:42 - Toshi MARUYAMA

- Assignee deleted (Toshi MARUYAMA)

I can not accept this patch.
Because "save" is for reducing DB access (#7146, #6013).

"save" need to update "extra_info" column of "repositories" table (r5762).

#3 - 2011-10-26 11:46 - Toshi MARUYAMA

> loading of large repositories to take a long time

This is only in case of after upgrading from 1.1 to 1.2 or new branch at first time (#8857).

#4 - 2011-10-26 17:53 - John Kubiatowicz

The problem is that the existing code prevented me from importing a large repository into redmine because it **timed out**. The problem, as I mentioned, is that every save causes an event which is observed by the plugin that I am using.

Why is putting the save at the end a bad thing? It would seem to be strictly higher performing, since it accesses the database much less... It appears that the extra_info hash is accumulated as the code runs through the loop, thus this will not lose information....

#5 - 2011-10-27 02:47 - Mischa The Evil

Follow-up on Specific question about git scm implementation: follow on...

#6 - 2011-10-27 02:48 - Toshi MARUYAMA

I add a comment of a "last_scmid" purpose at r7658.

For web server timeout, you can use "ruby script/runner 'Repository.fetch_changesets' -e production".
See http://www.redmine.org/issues/8857#note-15

For preventing "repositories" table update, you can split "extra_info" column to new table.

#7 - 2011-10-27 18:32 - John Kubiatowicz

Forgive me for being obtuse, but your comment doesn't seem to explain why you need to execute self.save in the loop over the revisions. Why can't it be at the end of the loop? It seems to work fine (been using it on a complex repository for a couple of days now). I'm getting the impression by your answers that you misunderstand my query -- I don't want to remove the save, just put it at the end (rather than looping over it).

As for executing it with script/runner, this doesn't work as shown by #8857 -- there is still so much traffic that the process doesn't complete. Presumably I don't want to hack your code to split a separate table (and not sure that does the right thing, since I believe that the plugin wants to know when the repository information has been updated).

#8 - 2011-10-27 18:58 - Toshi MARUYAMA

John Kubiatowicz wrote:

> Why can't it be at the end of the loop?

Because "self.save" saves "last_scmid".

If "last_scmid" does not save in case of time out,
next time, redmine reads all revisions that already been saved in database.

#9 - 2011-10-27 19:38 - Etienne Massip

Toshi MARUYAMA wrote:

> If "last_scmid" does not save in case of time out,
> next time, redmine reads all revisions that already been saved in database.

Actually, if it timed out, then there may be something wrong already, and we can expect this situation to be rare, so re-fetching revisions is not that harmful?

By "all revisions", you mean all revisions that hadn't been fetched before the call to #fetch_changesets, don't you?

What you're saying is that thanks to the save in the loop, subsequent calls to #fetch_changesets can success?

#10 - 2011-10-27 20:53 - John Kubiatowicz

Toshi MARUYAMA wrote:

> John Kubiatowicz wrote:
>
> > Why can't it be at the end of the loop?
>
> Because "self.save" saves "last_scmid".
>
> If "last_scmid" does not save in case of time out,
> next time, redmine reads all revisions that already been saved in database.

Hm... This is a failure mode that never occurred to me. Several things occur to me here:

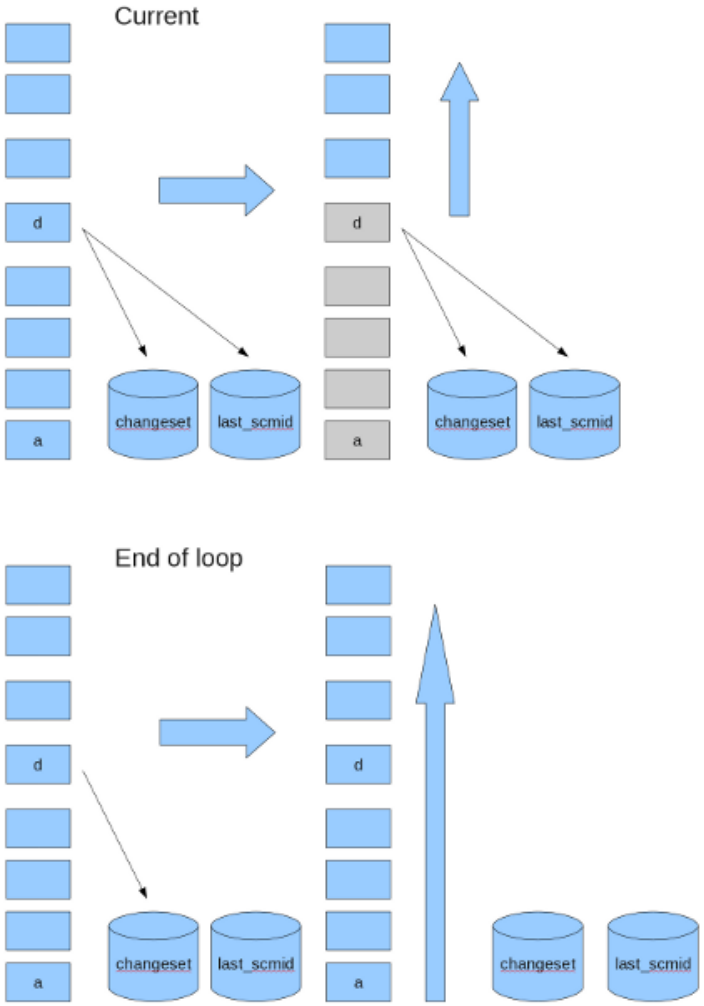
1) The database access presumably takes much longer than an individual loop (100x, 1000x?); consequently, you would seem to be making a timeout problem much worse by saving every loop -- even in a vanilla redmine without plugins. At minimum, you should try to figure out the blow-up factor and save only every n loops, where n is the expected ratio of time to save to the database vs parse the next revision ID from the output of git....
2) I would agree with Etienne here. This seems like a case where something is wrong already if you get an actual timeout. Wouldn't this rare case be better handled by the "ruby script/runner 'Repository.fetch_changesets' -e production" solution you suggested to me, rather than impacting the normal case by orders of magnitude (see #1) in time?

#11 - 2011-10-28 00:26 - Toshi MARUYAMA

- File git-1.png added

- File git-2.png added

- File git.odg added



#12 - 2011-10-28 11:59 - Etienne Massip

- File git_brch_tx.patch added

Still, by "all revisions", you mean all revisions that hadn't been fetched before the call to #fetch_changesets, don't you?

Would you prefer this one?

#13 - 2011-10-28 12:20 - Etienne Massip

- File git_brch_tx.patch added

Cleaner one. Actually, I'm not sure the transaction block here is worth it since the #find_by_name is called anyway for each rev, removing it would save some #save_revision calls? Maybe then it would be needed inside #save_revision?

#14 - 2011-10-28 12:27 - Etienne Massip

- File deleted (git_brch_tx.patch)

#15 - 2011-10-28 12:30 - Etienne Massip

- File git_brch_no_tx.patch added

What I meant.

#16 - 2011-10-28 13:45 - Toshi MARUYAMA

Subversion and Mercurial have a sequential revision number,
So, Redmine can know **last** saved revision number from **changesets** table.

Subversion: (1, 2, 3 ....)
source:tags/1.2.1/app/models/repository/subversion.rb#L54

Mercurial: (0, 1, 2 ....)
source:tags/1.2.1/app/models/repository/mercurial.rb#L128

But, Git does not have a sequential revision number.
And Git branch is the pointer to the specific revision.
So, Redmine can **not** know **last** saved revision from **changesets** table.

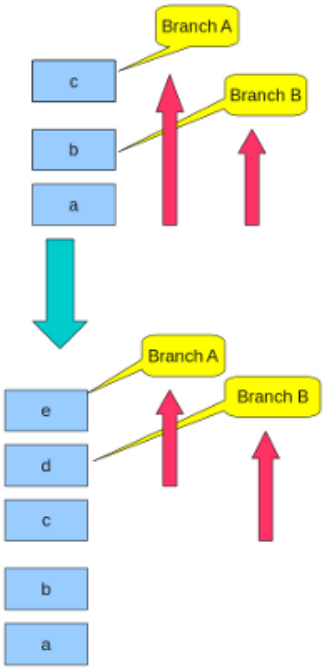
In order to know **last** saved revision per branch,
Redmine saves **last** saved revision id per branch to database.
Current implemention is **repositories** table.

So, saving "changeset" and "last saved revision id" need in transaction.

#17 - 2011-10-28 14:32 - Toshi MARUYAMA

- File git-branch.odg added

- File git-branch.png added



#18 - 2011-10-28 15:05 - Etienne Massip

While I find Toshi's method a bit paranoid, I understand his pov since scm.save would not be a bottleneck if this observer thing wasn't actually doing much after each call to #save.

The redmine_git_plugin actually extrapolates the fact that a Repository#save corresponds to the end of a revisions fetching operation.

What might be missing on RM side is maybe some hooks before and after calling Repository#fetch_changeset. The plugin could still use some RoR alias_method_chain technique, though (described in wiki in Plugin_Internals).

#19 - 2011-10-28 18:34 - John Kubiatowicz

Yes, I guess I can see this point of view -- although I still think it is optimizing for a failure case that shouldn't happen. Looking more carefully at this code, I believe that there is a bunch of DB traffic that happens per iteration already, when creating new changesets, which negates my performance argument from earlier.

I believe that one problem is that the plugin really only wants to know about major repository changes, not changeset alterations. Yes, I could wrap the fetch_changeset with something that disables the callback.

Question: instead of putting this information in the extra_info hash, why not use changeset entries, one for each branch. They could have the branch name for their identifier and the scmid for last_scmid. They could easily be distinguished from normal git changeset entries for which the identifier and scmid are equal. This would seem to work, no? (Although it is equivalent to the suggestion of putting last_scmid info in a separate table, which is perhaps the best solution...?)

As I think about this code, what happens if the last_scmid for a branch is nolonger on the branch? (i.e. because of, say "git commit --amend" and "git push -f" operations). It appears that "git log foo.bar" where foo is not an ancestor of bar just shows bar. The result would seem to miss the new (altered) history entries, since in this case it would be better to recognize this situation and set foo=>nil.

#20 - 2011-10-28 18:36 - John Kubiatowicz

Above, I meant "git log foo...bar", of course. In the case where foo is not an ancestor of bar, this only shows the bar log entry.

#21 - 2011-10-28 19:18 - Toshi MARUYAMA

John Kubiatowicz wrote:

> why not use changeset entries, one for each branch.

There were serious fetching performance problems.

- 0.9.0:
  - source:tags/0.9.0/app/models/repository/git.rb#L40
  - #4547
  - #4716
- 0.9.3:
  - source:tags/0.9.3/app/models/repository/git.rb#L40
  - #7146
  - https://www.chiliproject.org/issues/214
  - #6013

> (i.e. because of, say "git commit --amend" and "git push -f" operations).

There is test for this situation.
source:tags/1.2.1/test/unit/lib/redmine/scm/adapters/git_adapter_test.rb#L190

#22 - 2011-10-28 21:04 - John Kubiatowicz

Toshi MARUYAMA wrote:

> > (i.e. because of, say "git commit --amend" and "git push -f" operations).
>
> There is test for this situation.
> source:tags/1.2.1/test/unit/lib/redmine/scm/adapters/git_adapter_test.rb#L190

Actually, (ironically?) this tested behavior is exactly the wrong behavior for the code in git.rb. When the last_scm is not on the respective branch, one actually needs to reset from_scm=>nil and try again (so that you get a potentially large number of revisions, rather than no revisions).

#23 - 2011-10-29 01:04 - Toshi MARUYAMA

John Kubiatowicz wrote:

> one actually needs to reset from_scm=>nil and try again (so that you get a potentially large number of revisions, rather than no revisions).

As I described at issue 4455 note 28 , history editing is rare case on **shared** repository.
We agreed Redmine does not need care history editing for **shared** repository at issue 4455 note 150 .

#24 - 2012-03-16 17:29 - Toshi MARUYAMA

- Status changed from New to Closed

- Target version set to 1.4.0

r9144 fixed this issue.

#25 - 2012-03-17 00:11 - Toshi MARUYAMA

- Resolution set to Fixed

Files

git.patch	947 Bytes	2011-10-26	John Kubiatowicz
git-1.png	22 KB	2011-10-27	Toshi MARUYAMA
git-2.png	19.8 KB	2011-10-27	Toshi MARUYAMA

git.odg	10.9 KB	2011-10-27	Toshi MARUYAMA
git_brch_tx.patch	946 Bytes	2011-10-28	Etienne Massip
git_brch_no_tx.patch	1.49 KB	2011-10-28	Etienne Massip
git-branch.odg	10.8 KB	2011-10-28	Toshi MARUYAMA
git-branch.png	17.6 KB	2011-10-28	Toshi MARUYAMA