# Redmine - Defect #8857

## Git: Too long in fetching repositories after upgrade from 1.1 or new branch at first time

2011-07-20 10:53 - Paul Wilson

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 2011-07-20 |
| **Priority:** | Low | **Due date:** | |
| **Assignee:** | Toshi MARUYAMA | **% Done:** | 100% |
| **Category:** | SCM | **Estimated time:** | 0.00 hour |
| **Target version:** | 1.4.0 | | |
| **Resolution:** | Fixed | **Affected version:** | 1.2.0 |

**Description**

Our installation has been stable and using git repositories for well over one year now. Four months ago the upgrade from 1.1 (r5184) to 1.1.2 went without a hitch but this upgrade from 1.1.2 to 1.2.0 or 1.2.1 (I tried both revs with the same result) just isn't working.

Following the upgrade, selecting on the 'repository' tab in any project stops everything but nothing is added to the log. Symptoms are the same when run with either webrick or mongrel_rails.

Selecting on the 'repository' tab spawns a process:

```
git --git-dir <path to project's repository> log --no-color --encoding=UTF-8 --raw --date=iso --pretty=fuller --reverse
feature/12591_threadedMessage
```

but the process never finishes and Ruby consumes most of the CPU until the deamon is killed.

The git repository is local to this machine and I can execute the command interactively as the user that owns the redmine deamon and results are nearly immediate.

Prior to release 1.2.0, this was working flawlessly.

The problem wasn't immediately evident and we operated for nearly two full days until we became aware of it when someone tried to browse a repository. As the schema upgrade in 1.2 only added columns to some tables in support of the private issues feature, we were able to safely roll back the software version to 1.1.2 while keeping the upgraded database schema ad not losing two days of activity.

O/S info:

```
SUSE Linux Enterprise Server 10 (x86_64)
```

VERSION = 10
PATCHLEVEL = 1

MySQL versions:

```
local client version 5.0.26
```

remote server version 5.0.51a-community-log MySQL Community Edition

git version

```
1.7.0.5
```

RAILS_ENV=production ruby script/about

```
About your application's environment
```

```
Ruby version              1.8.6 (x86_64-linux)
RubyGems version          1.3.6
```

| | |
|---|---|
| Rack version | 1.1 |
| Rails version | 2.3.11 |
| Active Record version | 2.3.11 |
| Active Resource version | 2.3.11 |
| Action Mailer version | 2.3.11 |
| Active Support version | 2.3.11 |
| Application root | /opt/redmine-1.2.0 |
| Environment | production |
| Database adapter | mysql |
| Database schema version | 20110511000000 |

**Related issues:**

| | | |
|---|---|---|
| Related to Redmine - Defect #7146: Git adapter lost commits before 7 days fro... | **Closed** | **2010-12-21** |
| Related to Redmine - Defect #1435: slow display of 'repository' tab for huge ... | **Closed** | **2008-06-12** |
| Related to Redmine - Defect #3449: Redmine Takes Too Long On Large Mercurial ... | **Closed** | **2009-06-05** |
| Related to Redmine - Defect #4547: git: Very high CPU usage for a long time w... | **Closed** | **2010-01-11** |
| Related to Redmine - Defect #4716: Git repository performance fall on parsing... | **Closed** | **2010-02-02** |
| Related to Redmine - Defect #6013: git tab,browsing, very slow -- even after ... | **Closed** | **2010-08-02** |
| Related to Redmine - Defect #9472: The git scm module causes an excess amount... | **Closed** | **2011-10-26** |
| Related to Redmine - Patch #10470: Efficiently process new git revisions in a... | **Closed** | |
| Has duplicate Redmine - Defect #8973: long git loop after upgrade to 1.2.1 | **Closed** | **2011-08-02** |

## Associated revisions

**Revision 8815 - 2012-02-09 00:38 - Toshi MARUYAMA**

scm: git: add includes and excludes options to lib revisions method (#8857)

**Revision 8816 - 2012-02-09 01:11 - Toshi MARUYAMA**

scm: git: add model method to get heads from extra_info branches hash (#8857)

**Revision 8817 - 2012-02-09 07:05 - Toshi MARUYAMA**

scm: git: add lib test to get master branch revisions from origin (#8857)

**Revision 8820 - 2012-02-09 09:20 - Toshi MARUYAMA**

scm: git: update test repository (#8857)

New disjointed history revisions are added.
This test repository has two origins.

**Revision 8821 - 2012-02-09 09:34 - Toshi MARUYAMA**

scm: git: add test of get revisions from disjointed histories to unit lib test (#8857)

**Revision 8822 - 2012-02-09 16:07 - Toshi MARUYAMA**

scm: git: add explicit :excludes option to calling revision method in "test_revisions_includes_master_two_revs_from_origin" of unit lib test (#8857)

**Revision 8837 - 2012-02-10 23:56 - Toshi MARUYAMA**

scm: git: raise exception if "git log" is error and block is given in lib revision method (#8857)

**Revision 8839 - 2012-02-11 06:42 - Toshi MARUYAMA**

scm: git: call "git log" only once instead of per branch in fetching revisions (#8857)

**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 9142 - 2012-03-07 06:57 - Toshi MARUYAMA**

scm: git: backout r8839 (#8857)

call "git log" only once instead of per branch 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 - 2011-07-20 11:21 - Etienne Massip**

*- Category set to SCM*

**#2 - 2011-07-20 11:23 - Etienne Massip**

Did you try to run the git command line in a shell ?

**#3 - 2011-07-20 11:36 - Paul Wilson**

I wrote above:

> "The git repository is local to this machine and I can execute the command interactively as the user that owns the redmine deamon and results are nearly immediate."

The daemon is normally as the machine's apache user and 'Yes,' I can successfully run the command in a shell on this host as the apache user.

**#4 - 2011-07-20 11:42 - Etienne Massip**

Ok, sorry for misreading.

**#5 - 2011-07-20 14:49 - pasquale [:dedalus]**

I have similar problem upgrading to 1.2.1: my mercurial repros are unreachable (error 500): maybe it's related to new default utf-8 encoding? I suppose that I should change some things in config/configuration.yml because in administration panel--> repositories, I have this message: "You can configure your scm commands in config/configuration.yml. Please restart the application after editing it".

**#6 - 2011-07-20 14:53 - pasquale [:dedalus]**

dedalus - wrote:

> I suppose that I should change some things in config/configuration.yml

I suppose is the path to mercurial binary and python libary

**#7 - 2011-07-20 15:49 - Etienne Massip**

*- Assignee set to Toshi MARUYAMA*

**#8 - 2011-07-20 22:37 - Toshi MARUYAMA**

*- Subject changed from Cannot access got repositories after upgrade from 1.1.2 (r5184) to 1.2.0 (r6069) or 1.2.1 (r6294) to Cannot access got repositories after upgrade from 1.1 to 1.2*

*- Priority changed from High to Normal*

**#9 - 2011-07-20 22:43 - Toshi MARUYAMA**

*- Subject changed from Cannot access got repositories after upgrade from 1.1 to 1.2 to Git: Too Long to get repositories after upgrade from 1.1 to 1.2 or new branch*

**#10 - 2011-07-20 22:48 - Toshi MARUYAMA**

*- Subject changed from Git: Too Long to get repositories after upgrade from 1.1 to 1.2 or new branch to Git: Too Long to get repositories after upgrade from 1.1 to 1.2 or new branch at first time*

**#11 - 2011-07-20 23:45 - Toshi MARUYAMA**

*- Assignee deleted (Toshi MARUYAMA)*

*- Priority changed from Normal to Low*

> Selecting on the 'repository' tab spawns a process:
>
> > git --git-dir <path to project's repository> log --no-color --encoding=UTF-8 --raw --date=iso --pretty=fuller --reverse feature/12591_threadedMessage

This is intended behavior.

Redmine Git adapter had big problems in fetching revisions.

- From 0.9.0 to 0.9.2: #4547, #4716
- From 0.9.3 to 1.1: #7146

To resolve these problem, I added new column at database source:tags/1.2.1/app/models/repository/git.rb#L131 .
**last_scmid** stores last revision id per branch.

At first time upgrading Redmine 1.1 to 1.2, last_scmid is **nil**.
So, Redmine calls "git log --reverse master" at source:tags/1.2.1/lib/redmine/scm/adapters/git_adapter.rb#L189
And Redmine set last_scmid **4a4a71349a45bdc8a55071e535bc0a8b9c02a5ee** for master branch.

```
$ git log --reverse master

commit d19142e20949d442c20275e467f3d39c33a21f03
Author: Jean-Philippe Lang
Date:   Wed Jun 28 17:26:26 2006 +0000

    Initial setup

    git-svn-id: http://redmine.rubyforge.org/svn/trunk@1 e93f8b46-1217-0410-a6f0-8f06a7374b81

commit 73e0b8f8b3c9350018a7d8fda143dd9b8aa6f091
Author: Jean-Philippe Lang
Date:   Wed Jun 28 18:09:47 2006 +0000

    Initial import.

    git-svn-id: http://redmine.rubyforge.org/svn/trunk@3 e93f8b46-1217-0410-a6f0-8f06a7374b81

.
.
.

commit c7c062a981f14b256eb19a68664b08f21e5e8cc7
Author: Jean-Philippe Lang
Date:   Sat Jul 9 20:46:44 2011 +0000

    Cleanup select filter tag.

    git-svn-id: svn+ssh://rubyforge.org/var/svn/redmine/trunk@6215 e93f8b46-1217-0410-a6f0-8f06a7374b81

commit 4a4a71349a45bdc8a55071e535bc0a8b9c02a5ee
Author: Jean-Philippe Lang
Date:   Sat Jul 9 21:34:35 2011 +0000

    Fixes "less than", "greater than" filters on custom fields with postgres (#6180).
    git-svn-id: svn+ssh://rubyforge.org/var/svn/redmine/trunk@6216 e93f8b46-1217-0410-a6f0-8f06a7374b81
```

At next time, Redmine calls "git log --reverse 4a4a71349a45bdc8..master"

```
$ git log --reverse 4a4a71349a45bdc8..master

commit 932d4cdfead379e24934df6530f4d98abcfab18e
Author: Jean-Philippe Lang
Date:   Sun Jul 10 08:00:25 2011 +0000

    Adds "between" operator for numeric filters (#6180).

    git-svn-id: svn+ssh://rubyforge.org/var/svn/redmine/trunk@6217 e93f8b46-1217-0410-a6f0-8f06a7374b81

commit cab469836143ee992f6187fadc4d66fcaaf741bf
Author: Toshi MARUYAMA
Date:   Sun Jul 10 11:07:09 2011 +0000

    remove trailing white-spaces from fetch_changesets.rake.

    git-svn-id: svn+ssh://rubyforge.org/var/svn/redmine/trunk@6218 e93f8b46-1217-0410-a6f0-8f06a7374b81
```

As I described at source:tags/1.2.1/app/models/repository/git.rb#L95 ,
there is no way to prevent this issue.

#### #12 - 2011-07-20 23:50 - Toshi MARUYAMA

*- Subject changed from Git: Too Long to get repositories after upgrade from 1.1 to 1.2 or new branch at first time to Git: Too Long to fetch repositories after upgrade from 1.1 to 1.2 or new branch at first time*

#### #13 - 2011-07-21 00:26 - Toshi MARUYAMA

*- Subject changed from Git: Too Long to fetch repositories after upgrade from 1.1 to 1.2 or new branch at first time to Git: Too Long in fetching repositories after upgrade from 1.1 to 1.2 or new branch at first time*

#### #14 - 2011-07-21 04:00 - Paul Wilson

I believe I can confirm your conclusions. In my staging environment on a project with a small number of commits in its repository, I was finally patient enough to wait for the process to complete.

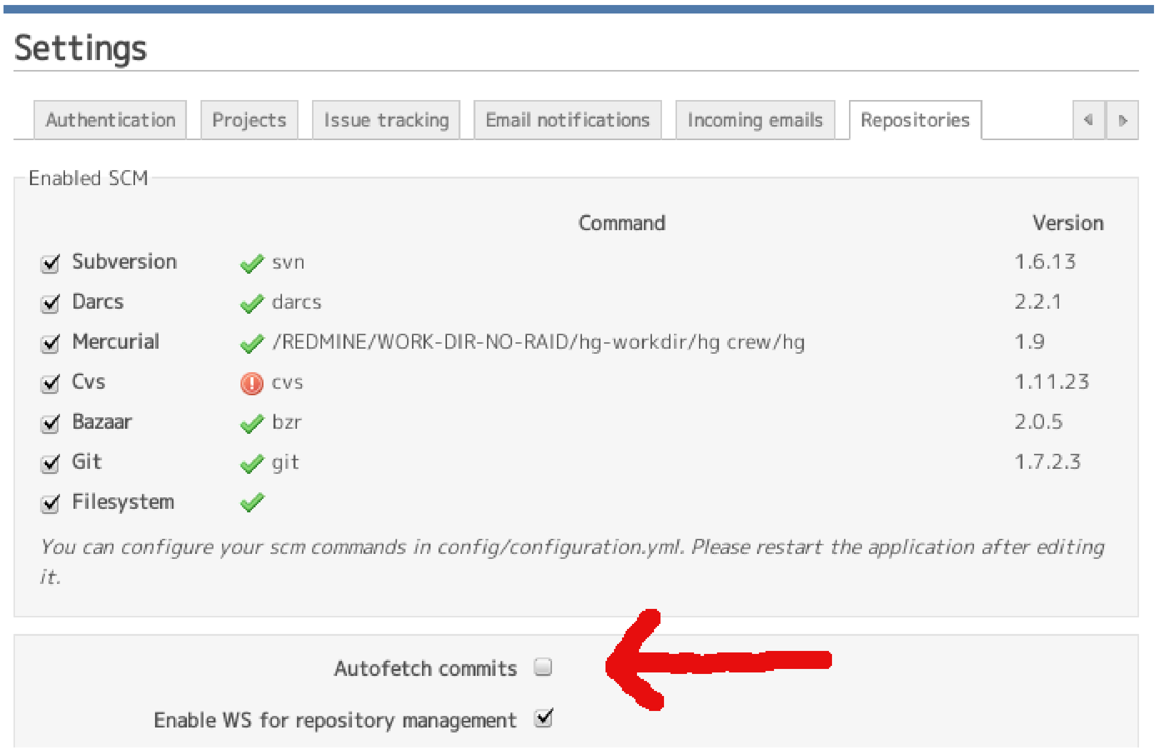
Our usage load permits it so I run our redmine single-threaded in mongrel so this long, load-time behavior causes unacceptable delays for everyone. Can this be done offline? Or is there perhaps any advantage to be gained from clearing and reloading the changesets or even dropping and reloading the repository?

#### #15 - 2011-07-21 04:34 - Toshi MARUYAMA

*- File repo-setting.png added*

Paul Wilson wrote:

> Can this be done offline?


Settings

Authentication | Projects | Issue tracking | Email notifications | Incoming emails | Repositories

Enabled SCM

|  | Command | Version |
|---|---|---|
| ✓ Subversion | ✓ svn | 1.6.13 |
| ✓ Darcs | ✓ darcs | 2.2.1 |
| ✓ Mercurial | ✓ /REDMINE/WORK-DIR-NO-RAID/hg-workdir/hg crew/hg | 1.9 |
| ✓ Cvs | ! cvs | 1.11.23 |
| ✓ Bazaar | ✓ bzr | 2.0.5 |
| ✓ Git | ✓ git | 1.7.2.3 |
| ✓ Filesystem | ✓ |  |

*You can configure your scm commands in config/configuration.yml. Please restart the application after editing it.*

Autofetch commits ☐

Enable WS for repository management ☑

```
$ /usr/bin/wget -q -O /dev/null --timeout=0 http://localhost/sys/fetch_changesets?key=xxxxx
```

Or

```
$ ruby script/runner "Repository.fetch_changesets" -e production
```

**#16 - 2011-07-21 05:14 - Paul Wilson**

Sorry Toshi but please clarify. It appears you are recommending to **de-select** Autofetch commits but must I then manually fetch commits somehow using cron or a hook?

**#17 - 2011-07-21 05:28 - Toshi MARUYAMA**

Paul Wilson wrote:

> Sorry Toshi but please clarify. It appears you are recommending to **de-select** Autofetch commits but must I then manually fetch commits somehow using cron or a hook?

Yes.

**#18 - 2011-07-21 08:56 - Felix Schäfer**

Paul Wilson wrote:

> Sorry Toshi but please clarify. It appears you are recommending to **de-select** Autofetch commits but must I then manually fetch commits somehow using cron or a hook?

If you have only one web process, that will be the way to go. Ruby is currently mostly single-threaded, scaling is achieved by adding processes, not threads, thus if you have only one process and auto-fetch on, auto-fetch will block this process until you are done. Please also note that the wget solution proposed by Toshi will also block the web thread as it calls it, that might happen when you're not using the website though.

The better solution for you would probably be to use either a cron or a post-commit (svn) or post-receive (git) hook to call the rake task or the runner as described in the FAQ. Those can also be adapted to fetch commits for exactly the right project instead of for all, do tell if you need help with that.

**#19 - 2011-07-21 08:58 - Paul Wilson**

Perhaps I am overlooking something:

Using my staging environment with it's own separate database, I managed to get one project's changesets updated and was finally able to normally browse this project's repository metadata after the upgrade.

I have de-selected Autofetching as advised.

I then made a commit to this project's repository followed by a http GET from <MyURL>/sys/fetch_changesets?id=<project identifier>&key=<service API key>. The GET took a bit less that the initial call during the upgrade but far, far longer than experienced pre-revision 1.2. Pre-revision 1.2 using

Autofetch, changesets were updated in nearly real-time. Now we are looking at huge lapses and a genuine degradation in useability.

Git hasn't changed, only the redmine git adapter has. What is the intended pay-back of the changes to the git adapter? As it stands now using git as the repository, the hit we are taking with the useability is unacceptable.

**#20 - 2011-07-21 09:11 - Toshi MARUYAMA**

Paul Wilson wrote:

> Git hasn't changed, only the redmine git adapter has. What is the intended pay-back of the changes to the git adapter? As it stands now using git as the repository, the hit we are taking with the useability is unacceptable.

Redmine Git adapter from 0.9.3 to 1.1 had serious problems.

- #7146
- #6013

#6013 is reading all 7 days revisions **every time**.
I can't accept this behavior.

This issue is **only first time** in upgrading to 1.2 or new branch.
Fetching at next time is very quickly.
And there is no losing revisions.

**#21 - 2011-07-21 10:27 - Paul Wilson**

Felix Schäfer wrote:

> The better solution for you would probably be to use either a cron or a post-commit (svn) or post-receive (git) hook to call the rake task or the runner as described in the FAQ. Those can also be adapted to fetch commits for exactly the right project instead of for all, do tell if you need help with that.

I will take you up on your offer to help with fetching commits for separate projects rather than for all.

Our redmine host machine is also the git repositories host and the path to the repository is local and references the actual repository, not a Git bare repository. I am thinking of a hook calling 'ruby $RedmineEnvironment/script/runner "Repository.fetch_changesets" -e production' but I would also like to identify the project when fetching the changesets. How do I identify the specific project?

And also, since this is an actual repository and not a bare clone, what is the better hook to make the call, post-receive or post-commit?

**#22 - 2011-07-21 10:39 - Paul Wilson**

One more thing I need to clarify that prefaces my last questions:

Once the upgrade has been completed, can I reinstate the Autofetch function or does this change to the redmine Git adapter necessitate leaving it disabled and fetching commits by other means?

**#23 - 2011-07-21 11:19 - Felix Schäfer**

Paul Wilson wrote:

> Once the upgrade has been completed, can I reinstate the Autofetch function or does this change to the redmine Git adapter necessitate leaving it disabled and fetching commits by other means?

If you add the post-receive hook as a standard to all your repos, you don't need the auto-fetch.

**#24 - 2011-07-21 11:34 - Paul Wilson**

So, how do I identify the specific project in the hooks call to script/runner? Is the project identifier passed as a parameter?

**#25 - 2011-07-21 11:36 - Felix Schäfer**

Paul Wilson wrote:

> So, how do I identify the specific project in the hooks call to script/runner? Is the project identifier passed as a parameter?

Just give me minute, pretty swamped here :-)

**#26 - 2011-07-21 11:37 - Etienne Massip**

I think you can't with the rake way ?

Check RedmineRepositories for details.

**#27 - 2011-07-21 12:04 - Felix Schäfer**

Paul Wilson wrote:

> Our redmine host machine is also the git repositories host and the path to the repository is local and references the actual repository, not a Git bare repository. I am thinking of a hook calling 'ruby $RedmineEnvironment/script/runner "Repository.fetch_changesets" -e production' but I would also like to identify the project when fetching the changesets. How do I identify the specific project?
>
> And also, since this is an actual repository and not a bare clone, what is the better hook to make the call, post-receive or post-commit?

I have mine in post-receive, I believe it's where stuff like that is supposed to go.

Okay, regarding the rake thing: Etienne is right that  the rake task updates all repositories, but the runner can be made to take an argument/environment variable. Something like that should work and not balk on unknown identifiers or projects without a repository:

ruby /somewhere/script/runner "Project.find_by_identifier(ENV['PROJECT_IDENTIFIER']).try(:repository).try(:fetch_changesets)"

For this you need the PROJECT_IDENTIFIER environment variable set correctly.

You can get the identifier from the path to your repo if you have named your repos accordingly to the identifiers of your project. IIRC, the git hooks are executed in the hooks directory of your repository. As my repos all reside in the same place, and thus are at the same depth, I get the identifier from echo $PWD | awk -F '/' '{print $5;}' (my repos are in /var/git/somehost/someidentifier), YMMV.

OK, I think with all this you should be able to soup up your own tailored solution, if you still have questions, do ask.

**#28 - 2011-07-21 22:30 - Etienne Massip**

*- Status changed from New to Closed*

*- Resolution set to Invalid*

**#29 - 2011-08-25 09:54 - Jonas Juselius**

(For more background information, see my forum post http://www.redmine.org/boards/2/topics/25785).

It seems fetch_changesets is really broken in 1.2.1. I previously wrote that the first time I ran fetch_changesets, it ran for a few hours. Well I was probably wrong, I think it crashed because when it finished, there were still a lot of missing commits in the repository browser. I tried to check everything, raked the db and plugins, and restarted fetch_changesets. It has now been running for more than 24 hours, and is still running. It seems fetch_changesets is stuck in an infinite loop, because it's spewing out the exact same git messages, over and over again:

```
remote: Counting objects: 2302, done.
remote: Compressing objects: 100% (1885/1885), done.
remote: Total 2302 (delta 888), reused 1079 (delta 412)
Receiving objects: 100% (2302/2302), 270.15 KiB, done.
Resolving deltas: 100% (888/888), done.
```

**#30 - 2011-08-25 11:08 - Paul Wilson**

Jonas, if you are watching this issue, your issues seem very similar to what we originally experienced.

I summarize the key changes we needed to make here:

- As Toshi indicated (http://www.redmine.org/issues/8857#note-15), Autofetch commits needs to be disabled in the general settings for repositories
- I choose to update the changesets via a hook on each project's commit and it seems to be working quite well for us. I included this code in the post-receive hook for each project's repository that uses redmine:
  (In my example on our Linux server, the 'git' user owns all repositories and redmine is run by the apache user 'wwwrun.' The repositories and redmine happen are conveniently on the same machine and user 'git' has sudo rights to run commands as the apache user with NOPASSWD. The only variation from one hook script to the next is the project identifier which is fixed in redmine for each project. Thanks go to Felix for how to call the fetch_changesets for a specific project.)

  PROJECT_IDENTIFIER="<the project's redmine identifier goes here>"

RAILS_ENV="production"
sudo -u wwwrun env PROJECT_IDENTIFIER=$PROJECT_IDENTIFIER RAILS_ENV=$RAILS_ENV \
/usr/bin/ruby /srv/www/redmine/script/runner \
"Project.find_by_identifier(ENV['PROJECT_IDENTIFIER']).try(:repository).try(:fetch_changesets)" &

- Run the initial changeset upgrade offline the first time as it can take quite a while to complete and you will probably not want several process competing for a full update to each projects changeset information at the same time.
- Lastly, I have also updated the git-adapter with this version:

> http://www.redmine.org/projects/redmine/repository/entry/tags/1.2.1/lib/redmine/scm/adapters/git_adapter.rb#L189. It was the only patch I found necessary.

It may take some time to get everything in sync but don't give up. It does work.

#### #31 - 2011-08-25 11:24 - Jonas Juselius

Thank you for the reply! I have disabled autofetch, and I am running fetch_changesets offline. The problem is that it never finishes, it just loops and seems to do nothing. I'll test the new git_adapter.rb, but the diffs seem to be quite small.  After some more investigation, it seems the problem is related to gitosis: The infinite loop of git messages stems from the process of cloning the gitosis-admin.git repository. For some reason gitosis-admin gets cloned over and over and over again. As far as I can tell, it never gets past that stage.

#### #32 - 2011-08-25 15:22 - Jonas Juselius

I have now verified that it's the gitosis plugin which brakes fetch_changesets. I removed the plugin and fetch_changesets finished correctly in less than 60 seconds.

#### #33 - 2011-12-08 17:06 - Jeremy Bopp

*- File Optimize-Git-operations-for-new-branches.patch added*

*- File Alias-the-revision-and-scmid-properties-for-Git-Revi.patch added*

*- Status changed from Closed to Reopened*

Toshi MARUYAMA wrote:

> > Selecting on the 'repository' tab spawns a process:
> >
> > > git --git-dir <path to project's repository> log --no-color --encoding=UTF-8 --raw --date=iso --pretty=fuller --reverse feature/12591_threadedMessage
>
> This is intended behavior.
>
> Redmine Git adapter had big problems in fetching revisions.
>
> - From 0.9.0 to 0.9.2: #4547, #4716
> - From 0.9.3 to 1.1: #7146
>
> To resolve these problem, I added new column at database source:tags/1.2.1/app/models/repository/git.rb#L131 .
> **last_scmid** stores last revision id per branch.
>
> At first time upgrading Redmine 1.1 to 1.2, last_scmid is **nil**.
> So, Redmine calls "git log --reverse master" at source:tags/1.2.1/lib/redmine/scm/adapters/git_adapter.rb#L189
> And Redmine set last_scmid **4a4a71349a45bdc8a55071e535bc0a8b9c02a5ee** for master branch.
>
> [...]
>
> At next time, Redmine calls "git log --reverse 4a4a71349a45bdc8..master"
>
> [...]
>
> As I described at source:tags/1.2.1/app/models/repository/git.rb#L95 ,
> there is no way to prevent this issue.

Actually, preventing this issue is possible by excluding all of the last seen head revisions and their ancestors from the log output starting at the current head revisions.  I have included a patch generated by git format-patch that implements this behavior.  It should apply cleanly to revision 8125 of trunk.  A migration and updated tests are included.

With this change, pushing a new branch will only process the new revisions, if any, included on that branch since the last time changesets were fetched for the repository.  We need this at my company because we have a repository with over 20k revisions, and adding a new branch would take about 8 minutes of processing for Redmine, even if that branch did not yet contain any new commits.  Occasionally, we would have multiple independent pushes at nearly the same time which would cause the server to become overloaded and unresponsive for more than 30 minutes as the workers hammered the DB.

I have also attached another patch that will also help a bit.  There is a defect with the way the Revision object is instantiated in the git adapter and the way the fetch_changesets method checks for known revisions in the DB.  As instantiated prior to this patch, the scmid attribute is nil; however, the fetch_changesets method uses that attribute to look up revisions in the DB prior to trying to insert a new revision.  Obviously, that lookup will always fail to find anything by searching for a nil revision string.  The patch aliases the revision and scmid atrtibutes for git only.  This patch may be applied independently of the other, so I'll open a separated issue to address this particular defect if desired.

#### #34 - 2011-12-09 00:00 - Toshi MARUYAMA

*- Status changed from Reopened to 7*

*- Assignee set to Toshi MARUYAMA*

#### #35 - 2011-12-16 21:20 - Jeremy Bopp

*- File Optimize-Git-operations-for-new-branches.patch added*

I have updated the migration included in the main patch to account for a corner case where Redmine has not yet indexed a Git repository and thus has not yet defined the branches hash in the extra_info attribute.

#### #36 - 2011-12-16 23:43 - Toshi MARUYAMA

I reviewed note 33 and 35 patches.

- I don't want to use db migrations. It is too hard to test. It is disadvantage for non git users.
- What happens if repository has individual origin revisions?
  My github repository https://github.com/marutosi/redmine has two individual origin revisions.
  - Unofficial git mirror https://github.com/edavis10/redmine
    - https://github.com/marutosi/redmine/tree/bundler-20111019
  - Bitbucket Mercurial mirror https://bitbucket.org/redmine/redmine-trunk
    - https://github.com/marutosi/redmine/tree/bb-trunk

#### #37 - 2011-12-17 06:31 - Jeremy Bopp

Toshi MARUYAMA wrote:

> I reviewed note 33 and 35 patches.

Thank you for your time.

> - I don't want to use db migrations. It is too hard to test. It is disadvantage for non git users.

I'm not sure I understand the objection. Are you saying that there could be problems for users of Redmine who don't host git based projects? For non git users, it should be a no-op because the migration specifically seeks out only git repositories in the DB. In addition tests that depended upon the pre-migration state have all been updated to work with the post-migration state, and all pass as before.

The patch could be rewritten to avoid the need for the reformating of the extra_info attribute for git repositories; however, the current code simplifies things a bit partially because the superfluous branch names are no longer stored, only their revisions. To me the migration seems extremely low risk, especially for non git users of Redmine.

My original idea was actually to discard the extra_info thing entirely and add an attribute to the change sets that would enable them to be flagged as head revisions. The flag would be toggled as necessary as new heads were found and old heads were retired. This would have been a bigger change that **would** have affected users of other repository types since their change sets would have been modified to include this attribute even though it wasn't used. While it would have made atomic updates easier and cleaner for git in some ways, I didn't figure I could get that in at this time. :-)

> - What happens if repository has individual origin revisions?

By your example repository in github, I assume you mean a repository with two or more branches that have disjoint histories. I'll admit that I haven't tested this exact scenario, but the patch only uses standard git functionality to cull the list of revisions to process. There is no requirement in git that a repository have only a single origin, so the filtering should work regardless. If you add a new branch that has a different origin commit than existing branches (a disjoint history), the patch will cause Redmine to traverse all of the history of that branch and then record its current head revision for future reference. It doesn't matter that the currently stored head revisions are not able to filter any of the revisions in the new branch. That won't trigger an error, if that was your concern.

#### #38 - 2012-01-27 17:55 - Jeremy Bopp

*- Assignee deleted (Toshi MARUYAMA)*

Is there anything I need to do to move this forward?

#### #39 - 2012-01-27 22:51 - Toshi MARUYAMA

*- Assignee set to Toshi MARUYAMA*

*- Resolution deleted (Invalid)*

#### #40 - 2012-02-02 17:28 - Ulrich Scheller

We have the same problem with our Redmine/git and several bigger projects. It is not that visible, because fetching happens in the background through post-receive hooks. But still it burns up a lot of performance, because the server is basically doing Repository.fetch_changesets all day long.

If I would use Jeremys patches, could they cause trouble with later upgrades? I would prefer to have an official bugfix, of course. Is there a way to help moving this forward?

#### #41 - 2012-02-02 17:46 - Jeremy Bopp

Ulrich Scheller wrote:

> We have the same problem with our Redmine/git and several bigger projects. It is not that visible, because fetching happens in the background through post-receive hooks. But still it burns up a lot of performance, because the server is basically doing Repository.fetch_changesets all day long.
>
> If I would use Jeremys patches, could they cause trouble with later upgrades? I would prefer to have an official bugfix, of course. Is there a way to help moving this forward?

Because the main patch for the performance problem requires a migration, it's probable that there could be problems with later upgrades, especially if any of them also change the way the extra_info structure stored in the DB works.  For the time being, we're running in the configuration you're considering at my company, so we won't be able to accept any changes to Redmine that don't somehow include my patch.

That means we'll have to rework the patch as necessary going forward, which is good for you since we'll post the changes here; however, it also means that we're necessarily going to drag our feet taking *any* Redmine updates until the patch is accepted and applied.  For you that means that the updated patches will probably be delayed, possibly indefinitely. :-/

#### #42 - 2012-02-11 08:18 - Toshi MARUYAMA

*- Subject changed from Git: Too Long in fetching repositories after upgrade from 1.1 to 1.2 or new branch at first time to Git: Too Long in fetching repositories after upgrade from 1.1 or new branch at first time*

*- Target version set to 1.4.0*

*- % Done changed from 0 to 100*

#### #43 - 2012-03-01 16:06 - Steffen Zieger

Jonas Juselius wrote:

> I have now verified that it's the gitosis plugin which brakes fetch_changesets. I removed the plugin and fetch_changesets finished correctly in less than 60 seconds.

I've experienced the same issue. I'm running Redmine 1.3.1 with a working gitosis plugin: https://github.com/saz/redmine_gitosis

Altough gitosis is working, fetching changes takes more than 5 minutes. I hope, this bug will be fixed soon.
Anyway. Thanks for all the work! Redmine is really great.

#### #44 - 2012-03-06 18:26 - Gergely Fábián

*- File 0001-Performance-improvements-for-git-repo-parsing.patch added*

Hi everyone,

My company also experienced problems with git repo parsing after migration from 1.1 to 1.2/1.3. For quite a long time this was a reason why not to do the upgrade in production. However now finally I managed to put together a patch that solves the problem for us (returns 1.1's performance).
This doesn't contain any migrations, nor changes the way how (repository) data is saved to the database.
The main problem with 1.3's parsing is, that it takes terribly long for an unparsed repo, especially if it contains several branches. Our scenario was the following: we have a master branch with 6300+ commits, and several other branches that are merged/rebased from this one. So 1.3's parsing needs to check on these commits as many times as they are "replicated" on other branches. This increases parsing time by *N, if N is the number of branches.
With 1.1 parsing time for our repo was 15-20min. With 1.3 it was 1.4 hours. This is significant difference.
Once I realized the differences on how 1.1 does the repo parsing, I tried to apply something similar for 1.3.

So what I made:

1. First, I create an array, that I use to store the commit ids that I already added for any branches. When I start checking a new branch I first remove those revisions, that I've already added for other branches. This increases performance from the second branch.
2. Second, I modified the way how revision existence is checked in the database. It's really low performance to run thousands of SELECTs for each revision. I guess this causes the really high db load (if I'm right this is Defect #9472).
   I changed it to go through all revisions (that remained after nr1), group them in sets of 100s (could be even 1000s I think), and query these sets of 100s from the db. This is a 100x less number of selects. I could have done it in one query, however my experience with other web application frameworks made me to make the grouping, to ensure we won't use too much memory.
3. Related to the above there is a tiny change on how the last scmid is saved for a branch.

With the above changes repo parsing is back to 18min, which seems to be ok in our case.
To help testing I've also put a script to the patch that fetches the changesets for a given repository.

```
ruby script/repository_fetch_changesets [id|"last"]
```

Try running it with a newly created repository, with one that was already parsed (but unchanged), and with one that was parsed, but has some new commits since.

I hope this helps someone, and that eventually (at least partially) it will be included in upstream :)

**#45 - 2012-03-06 18:42 - Gergely Fábián**

One small note: I checked on the version committed to master, and it seems to be only a workaround, removing the separate parsing for all branches, in fact returning to what 1.1 was doing. In contrary my commit is fixing the problem (multiple branches + huge repo; at least for our case).
Looking forward to any opinions.
Thank you.

**#46 - 2012-03-06 23:51 - Toshi MARUYAMA**

*- % Done changed from 100 to 50*

**#47 - 2012-03-07 04:11 - oprimosr andeson**

-

**#48 - 2012-03-07 04:57 - Jeremy Bopp**

Hello, Gergely,

While the patches I have attached to this issue introduce a simple migration, they work by removing most of the load of revision processing from the DB by making git do the work of eliminating revisions that have already been seen and returning a list of revisions that are ready to be processed. This is both memory efficient and extremely fast.

I believe that the trunk (master branch in git) contains effectively the same code at this time, so new revisions are processed one time each, even if multiple new branches contain those revisions in their histories.

Now, there could be a potential memory consumption issue in the current code when dealing with updates to the repository that contain many, many revisions; such as when a large repository is added to Redmine for the first time. I believe that all of those revisions will be parsed and loaded into memory at once. Potentially, this could consume a great deal of memory, but I'm not sure it's likely to be significant realistically.

**#49 - 2012-03-07 09:49 - Toshi MARUYAMA**

*- % Done changed from 50 to 100*

**#50 - 2012-03-07 10:39 - Gergely Fábián**

Jeremy Bopp wrote:

> Hello, Gergely,
>
> While the patches I have attached to this issue introduce a simple migration, they work by removing most of the load of revision processing from the DB by making git do the work of eliminating revisions that have already been seen and returning a list of revisions that are ready to be processed. This is both memory efficient and extremely fast.

Sure, however as Toshi Maruyama highlighted previously, he doesn't want to use a solution that contains migrations.

> I believe that the trunk (master branch in git) contains effectively the same code at this time, so new revisions are processed one time each, even if multiple new branches contain those revisions in their histories.

Do you mean, that trunk and my solution is the same? I don't think so.
It's true, that the same commit for multiple branches won't be saved. However how it gets there could also be optimized. It runs a select (with limit 1) for each revision separately:

```
db_rev = find_changeset_by_name(rev.revision)
```

This is thousands of selects if a repository contains thousands of revisions. And it will be done over-and-over for each branch (even if the revisions are already in the db). It needs terribly much time just to check whether the revisions are already in the db.
The two optimization steps were that first save the revisions that were already parsed (and remove them right away once I get a new revisions array for another branch from git), and make the revision existence check faster even for the first branch (by grouping them into one query, and removing those found from the array).

> Now, there could be a potential memory consumption issue in the current code when dealing with updates to the repository that contain many, many revisions; such as when a large repository is added to Redmine for the first time. I believe that all of those revisions will be parsed and loaded into memory at once. Potentially, this could consume a great deal of memory, but I'm not sure it's likely to be significant realistically.

I'm not sure whether you meant the changes in my code, or generally loading in revisions from git. As for my addition (for remembering all revisions) I counted and one revision id takes 40 chars in git. It means 40 bytes. So the 'all_revisions' array for a repository with 30000 commits would consume

1.171 MiB. I guess, it's not so much. However if I was wrong, and indeed there is a memory problem here, then still by deleting the first part of optimization (removing the 'all_revisions' array, and it's usage), and leaving just the second part of it (grouping queries for revision existence) a lot of processing time would be saved.
I made this change, and run a test. Processing took again 18 min. So making the optimization in nr 1 doesn't give much speed improvements, eventually improvements in db load.
I guess also for loading revisions of one branch from git (in my code and current 1.3-stable) my calculation from above applies. Taking 1.1MiB for each branch (the array is recreated, so this doesn't add up) is not so much. The optimization in nr 1 may cause additional memory problems, just if there are several branches with really many (distinct) commits.

I just noticed, that trunk was updated.
Toshi, thank you for putting my changes in! :)

**#51 - 2012-03-07 11:00 - Toshi MARUYAMA**

I dropped r8839, because it generates very log command line if repository has many branches.

**#52 - 2012-03-07 12:22 - rlretersde John**

-

**#53 - 2012-03-07 12:47 - Jeremy Bopp**

Toshi MARUYAMA wrote:

> I dropped r8839, because it generates very log command line if repository has many branches.

It's possible to pass in all of the head revisions to be processed via stdin to git and avoid the long command line.  Suppose you have a head revision 8a8162c74352c487e0dbfa813f3b97b38a80d0a4 and a revision that has already been processed e67035c2d52abeec5573e5b96fdf571152bd1bc4.  You can tell git to give you the revisions that have not been processed yet by doing something like the following (in bash for the sake of this example):

```
{ echo 8a8162c74352c487e0dbfa813f3b97b38a80d0a4; echo ^e67035c2d52abeec5573e5b96fdf571152bd1bc4; } | git log -
-stdin
```

Obviously, Redmine could directly write the revisions into the pipe rather than construct such a pipeline using bash.  An arbitrary number of revisions can be passed in this way.  Note also that these revisions can be written to git in any order.  The key is that new head revisions would be written without the leading carrot (^) while old head revisions would be written with it.

I implemented my original solution in order to minimize code changes, but it should be pretty easy to further adjust it to run git in this way instead.

**#54 - 2012-03-07 13:12 - Jeremy Bopp**

Gergely Fábián wrote:

> > While the patches I have attached to this issue introduce a simple migration, they work by removing most of the load of revision processing from the DB by making git do the work of eliminating revisions that have already been seen and returning a list of revisions that are ready to be processed.  This is both memory efficient and extremely fast.
>
> Sure, however as Toshi Maruyama highlighted previously, he doesn't want to use a solution that contains migrations.

As I mentioned in my comment on that point originally, the code could be written to avoid the migration; however, storing branch names rather than the head revisions is what we really need here since the branch names will move over time to point to different revisions.  If you only use the names, you always have to hammer the DB in order to find the revisions that you have already processed.  You seem to have made that method a bit more efficient, but the alternative was still better since it didn't require any DB hits at all after looking up the old head revisions.

> > I believe that the trunk (master branch in git) contains effectively the same code at this time, so new revisions are processed one time each, even if multiple new branches contain those revisions in their histories.
>
> Do you mean, that trunk and my solution is the same? I don't think so.

No, that's not what I meant.

> It's true, that the same commit for multiple branches won't be saved. However how it gets there could also be optimized. It runs a select (with limit 1) for each revision separately:
>
> [...]
>
> This is thousands of selects if a repository contains thousands of revisions. And it will be done over-and-over for each branch (even if the revisions are already in the db).

No, that is not the case with my patches because git is used to exclude all revisions that were already processed.  A new repository that already had thousands of revisions **would** run that command thousands of times, but only the first time the repository was processed and only once for each revision in the repository at that time.  The number of branches is immaterial.

> It needs terribly much time just to check whether the revisions are already in the db.
> The two optimization steps were that first save the revisions that were already parsed (and remove them right away once I get a new revisions array for another branch from git), and make the revision existence check faster even for the first branch (by grouping them into one query, and removing those found from the array).

Grouping the revisions into a single query might still be a good idea here.  I think the reason they were done individually was in order to avoid races in the case that the repository is being concurrently processed in more than 1 thread or process.  Imagine what might happen if two git pushes that happen at nearly the same time both trigger Redmine to re-index the repository.  They may both start loading in the same set of new revisions.  Batching them could be problematic in this case.

> > Now, there could be a potential memory consumption issue in the current code when dealing with updates to the repository that contain many, many revisions; such as when a large repository is added to Redmine for the first time.  I believe that all of those revisions will be parsed and loaded into memory at once.  Potentially, this could consume a great deal of memory, but I'm not sure it's likely to be significant realistically.
>
> I'm not sure whether you meant the changes in my code, or generally loading in revisions from git.

I'm speaking of generally loading from git.  As I mentioned, the total list of revisions is pruned by git based on what has already been processed; however, a new repository with thousands of commits will list **all** of its revisions since there is nothing to exclude yet.  I believe both our solutions are vulnerable to this issue because of the way the git log is processed; however, your solution may actually be more vulnerable simply because it does not use git to exclude revisions that have already been processed.  As the repository grows, the list of revisions returned by git log will also grow.  This is independent of the DB processing.

The only solution for this is to rework the git log processing more extensively in order to facilitate loading the revisions from the log in a pipeline rather than slurping them all up into a big array at once.  Honestly, though, I think that should be managed as a separate issue to this one.  It's probably not a real issue anyway in these days of cheap RAM.

Did you actually test the performance of the trunk of Redmine prior to your changes being committed?  I would like to see how the two solutions compare for your repository.

How does your solution perform for a new branch that introduces no new revisions?  Say you simply do the following:

```
git checkout -b new_branch master
git push origin new_branch
```

My solution is effectively instant to process this because git will return no revisions at all to Redmine in this case.  If new_branch introduced 5 new revisions, **only** those 5 revisions would be processed through the DB, so again, the processing would be effectively instant.

I suspect your solution on your repository would still take on the order of 18 minutes, but I would like to hear otherwise.

**#55 - 2012-03-07 14:59 - Gergely Fábián**

Jeremy Bopp wrote:

> Did you actually test the performance of the trunk of Redmine prior to your changes being committed?  I would like to see how the two solutions compare for your repository.

I just did. The trunk version before my changes ran in 32 minutes. I guess the difference is more significant if the number of branches grows (currently there are 28).

> I suspect your solution on your repository would still take on the order of 18 minutes, but I would like to hear otherwise.

It took 41 seconds after creating a new branch from master (with Rails bootstrap). It couldn't really take much more, because it checks in the db in an effective manner even for the first branch (among those which are completely new). So it applies also for this case.

**#56 - 2012-03-07 15:14 - Gergely Fábián**

Jeremy Bopp wrote:

> I think the reason they were done individually was in order to avoid races in the case that the repository is being concurrently processed in more than 1 thread or process. Imagine what might happen if two git pushes that happen at nearly the same time both trigger Redmine to re-index the repository. They may both start loading in the same set of new revisions. Batching them could be problematic in this case.

If this is a problem, then I'd suggest the following solution:

1. Take the trunk code before my changes
2. Apply my optimization for saving the last_scmid just once for each branch (not at each hunderd, however that may be good for concurrent processes), and for saving the already processed commits into the 'all_revisions' array (to not to process them for other branches)

Or the one-by-one check could be eventually returned (but just after with the grouped db checks we eliminated those that we for sure don't need to process).

Regarding your migration changes I'm not in the position to opinion them. I just needed a solution, that makes the code run in a bearable time, and what I posted at first made that (and it works for us).
That's all I could add to this topic.

**#57 - 2012-03-07 16:37 - epetrossiapi aifseng**

-

**#58 - 2012-03-07 18:14 - ndeshongn andeson**

-

**#59 - 2012-03-08 21:28 - Gergely Fábián**

*- File 0001-Git-repo-parsing-optimization-without-db-query-group.patch added*

Gergely Fábián wrote:

> Jeremy Bopp wrote:
>
> > I think the reason they were done individually was in order to avoid races in the case that the repository is being concurrently processed in more than 1 thread or process. Imagine what might happen if two git pushes that happen at nearly the same time both trigger Redmine to re-index the repository. They may both start loading in the same set of new revisions. Batching them could be problematic in this case.
>
> If this is a problem, then I'd suggest the following solution:
>
> 1. Take the trunk code before my changes
> 2. Apply my optimization for saving the last_scmid just once for each branch (not at each hunderd, however that may be good for concurrent processes), and for saving the already processed commits into the 'all_revisions' array (to not to process them for other branches)

In other words, remove the db query grouping.
I made a patch for this version. It applies on top of current trunk/master (where my changes are already committed; git: 036b81abfd552bb9df766986b1a3715e72a56564).
Tests show, that our repository is parsed in 18 minutes (still), and parsing a brand new copy of the master branch took 60s (compare to 40s with the grouped queries).
I'm posting this, as an optional solution if Toshi or others would think my original version may have problems with concurrency.
And I also feel the need to note, that indeed the trunk version just before any of my changes was already a lot better than 1.3-stable.
So, in any case, please feel free to choose the solution that you think will be the most appropriate. :)

**#60 - 2012-03-08 22:14 - clgroschmu billaa**

-

**#61 - 2012-03-08 22:32 - Jeremy Bopp**

I actually think the batching is a great idea in conjunction with the original git optimizations that eliminate revisions that have already been processed, but all the processing needs to be protected by a transaction, including the last bit where the processed head revisions are recorded.  I'm not enough of a DB expert to know this for certain, but I think the only ramification of doing this would be blocking subsequent attempts to update the list of revisions in the same and other repositories that try to run at the same time.  It should not prevent actions that read from the revisions table, and the delay introduced should only be large when initially indexing new repositories and therefor rare.

It's important to point out that eliminating the previously processed revisions using git itself is pretty critical.  For small repositories, the difference is not large, but we were seeing significant issues with the reprocessing of revisions that were being done in version 1.2 and now again in trunk.  As I mentioned in my first note in this issue, one of our repositories is over 20k revisions, and probably close to 99% of those revisions were redundantly processed for every new branch that was added even if the branch included no new revisions of its own.  This is a waste of resources that git is well positioned to eliminate.

If I understand correctly, this part of the patch was dropped only because it could generate long command lines, but there is a simple solution for that problem which I documented in note 53.  I would like to hear if there are other issues with using git to eliminate previously processed revisions.

**#62 - 2012-03-09 02:37 - meintzmanj aifseng**

-

**#63 - 2012-03-09 16:32 - akestomm aifseng**

-

#### #64 - 2012-03-09 16:34 - rosturad John

-

#### #65 - 2012-03-09 16:52 - tewksburclar andeson

-

#### #66 - 2012-03-09 18:55 - estrevelva andeson

-

#### #67 - 2012-03-11 14:22 - Toshi MARUYAMA

Jeremy Bopp wrote:

> If I understand correctly, this part of the patch was dropped only because it could generate long command lines, but there is a simple solution for that problem which I documented in note 53.  I would like to hear if there are other issues with using git to eliminate previously processed revisions.

Redmine supports Windows and JRuby.
Redmine uses IO.popen.
source:tags/1.3.2/lib/redmine/scm/adapters/abstract_adapter.rb#L231
IO.popen uses shell.
http://ruby-doc.org/core-1.9.3/IO.html#method-c-popen
It means IO.popen uses "cmd.exe" on Windows.
It is not sure that "cmd.exe" works "{ echo a8162c74352c487e0d; } | git log --stdin"

#### #68 - 2012-03-12 15:43 - Jeremy Bopp

Toshi MARUYAMA wrote:

> It is not sure that "cmd.exe" works "{ echo a8162c74352c487e0d; } | git log --stdin"

My original example was meant as a simple proof of concept under Bash. :-)  Here is one way to translate it into Ruby that should be cross platform:

```
log_contents = IO.popen("git log --stdin", "r+") do |pipe|
  pipe.puts("8a8162c74352c487e0dbfa813f3b97b38a80d0a4")
  pipe.puts("^e67035c2d52abeec5573e5b96fdf571152bd1bc4")
  pipe.close_write
  pipe.read
end

puts log_contents
```

Doing this with the scm_cmd method should be essentially the same.  The logic that puts the revisions on the command line should be moved into the block handed to scm_cmd, converted to writes to the pipe, and then the write end of the pipe should be closed before the parsing logic.  This should be a net of about 5 changed lines based on my original patch.

I hope this clarifies things, but please let me know if it does not.

#### #69 - 2012-03-12 16:26 - Toshi MARUYAMA

Jeremy Bopp wrote:

> I hope this clarifies things, but please let me know if it does not.

I welcome your new patch.
But, Redmine 1.4.0 will be released on 2012-04-01
and feature freeze until Redmine 2.0 (Rails 3).
I want to close this issue for Redmine 1.4.0.

#### #70 - 2012-03-12 16:34 - Etienne Massip

Just a new dumb question from me: wouldn't the use of Rugged be a good solution for performance?

#### #71 - 2012-03-12 16:57 - Jeremy Bopp

Toshi MARUYAMA wrote:

> I welcome your new patch.
> But, Redmine 1.4.0 will be released on 2012-04-01
> and feature freeze until Redmine 2.0 (Rails 3).
> I want to close this issue for Redmine 1.4.0.

I'll try to attach a new patch to this issue in the next few days.

Etienne Massip wrote:

> Just a new dumb question from me: wouldn't the use of Rugged be a good solution for performance?

Something like that could be good in theory since it would avoid the trouble of parsing the output from git; however, it would at least have to be cross platform.  The big problem on the Windows side would be installing the gem and the library it depends on since both require compilation of native code.

**#72 - 2012-03-12 17:44 - Etienne Massip**

Jeremy Bopp wrote:

> (...) it would at least have to be cross platform.  The big problem on the Windows side would be installing the gem and the library it depends on since both require compilation of native code.

You're right, that's the point of https://github.com/libgit2/rugged/issues/43.

**#73 - 2012-03-13 16:16 - Jeremy Bopp**

I have a questions about this segment of code from the self.shellout method in lib/redmine/scm/adapters/abstract_adapter.rb:

```
if RUBY_VERSION < '1.9'
  mode = "r+"
else
  mode = "r+:ASCII-8BIT"
end
IO.popen(cmd, mode) do |io|
  io.close_write
  block.call(io) if block_given?
end
```

No matter the Ruby version, the pipe is opened in read-write mode (the r+ part of mode); however, the write end of the pipe is immediately closed upon entry into the block.  The changes I need to make in order to send the revisions to git via stdin require that the write end of the pipe be left open. Why is the pipe being closed like this?  Would it be safe to remove that io.close_write call?  I suspect that there are dependencies that require the write end of the pipe to be closed, so the only good alternative then would be to make an optional mode setting that could be passed in that would force the write end of the pipe to be left open.

BTW, that Ruby version check is extremely brittle.  It will fail to function correctly if RUBY_VERSION is '1.10' for instance.  While the next Ruby version **should be** 2.0, this check excludes the possibility of another interim version.

**#74 - 2012-03-13 22:35 - Toshi MARUYAMA**

Jeremy Bopp wrote:

> BTW, that Ruby version check is extremely brittle.  It will fail to function correctly if RUBY_VERSION is '1.10' for instance.  While the next Ruby version **should be** 2.0, this check excludes the possibility of another interim version.

Next Ruby version is 2.0.
https://github.com/ruby/ruby/commit/6b8d4ab840b2d76d356ba30dbccfef4f5fd10767

**#75 - 2012-03-13 22:46 - Jeremy Bopp**

Toshi MARUYAMA wrote:

> Jeremy Bopp wrote:
>
> > BTW, that Ruby version check is extremely brittle.  It will fail to function correctly if RUBY_VERSION is '1.10' for instance.  While the next Ruby version **should be** 2.0, this check excludes the possibility of another interim version.
>
> Next Ruby version is 2.0.
> https://github.com/ruby/ruby/commit/6b8d4ab840b2d76d356ba30dbccfef4f5fd10767

Yes, I understand that is the current plan at any rate, but this is still an extremely brittle way to check for functionality you need.  It would be far safer and more concise to do something like this instead:

```
mode = "r+"
IO.popen(cmd, mode) do |io|
  io.set_encoding("ASCII-8BIT") if io.respond_to?(:set_encoding)
  io.close_write
  block.call(io) if block_given?
end
```

Barring incompatible or broken implementations of IO#set_encoding in future Ruby versions or alternative Ruby implementations, this will work.

All that said, my primary question remains unanswered:

1. Why is the pipe opened in read-write mode only to immediately have the write end of the pipe closed?

I already answered my secondary question and have started to hack in a minimally invasive change to allow the pipe to be left open for writing when requested.

#### #76 - 2012-03-15 23:59 - Jeremy Bopp

*- File 0001-Pass-revisions-to-git-log-via-stdin.patch added*

*- File 0002-Process-new-git-revisions-all-at-once-rather-than-pe.patch added*

Here are 2 more patches that apply cleanly to the trunk at revision r9240.  The first modifies the usage of git-log to pass all revision arguments via stdin rather than on the command line.  This patch can be applied without the second patch if desirable, as it should not change the functionality exposed by the revisions method that uses git-log.  Doing this prepares the way for passing large numbers of revisions to git-log without overflowing the command buffer.

However, in order to support this new behavior, the shellout method had to be slightly modified so that the write end of the pipe it creates is left open upon request.  That change could potentially affect other consumers of that method, but I doubt it will.  Running the full scm test suite would be a good idea just in case though.  I only had time to test git functionality myself.

The second patch builds upon the first.  It processes all revisions in a single batch that are newly introduced since the last time the repository was processed.  Each revision in the batch is processed exactly once.  Disjoint branch histories and branch rewrites are supported.

All processing, including updating the last processed heads, occurs within a single transaction in order to ensure integrity of the data in case of concurrent attempts to update the repository.  This transaction could potentially block updates for other repositories hosted in the same Redmine instance; however, normal operation of git repositories should rarely introduce so many new revisions as to hold this transaction open for very long.  An initial import of a large repository on the order of thousands of commits would likely be the only realistic operation that could be a problem.  Given the infrequency of that, it is safe to document that such an import should be scheduled for server downtime.

Importantly, due to the resistance toward introducing a migration in my first patch set, this patch does not include any migrations.  A little extra processing is required to maintain the branch name to head revision relationship for every transaction, but this should be negligible.  I'll happily introduce another patch on top of this one though in order to do this head processing in a cleaner way that would require a migration.  Just let me know if you would take it.

#### #77 - 2012-03-16 01:04 - Toshi MARUYAMA

*- % Done changed from 100 to 70*

#### #78 - 2012-03-16 15:53 - Toshi MARUYAMA

Note 76 patches do not work on Windows.

```
$ ruby --version
ruby 1.9.3p0 (2011-10-30) [i386-mingw32]

  6) Error:
test_revisions_disjointed_histories_revisions(GitAdapterTest):
Redmine::Scm::Adapters::CommandFailed: git log error: git exited with non-zero status: 128
    R:/work/hg-workdir/REDMINE/my-work/lib/redmine/scm/adapters/git_adapter.rb:321:in `rescue in revisions'
    R:/work/hg-workdir/REDMINE/my-work/lib/redmine/scm/adapters/git_adapter.rb:211:in `revisions'
    test/unit/lib/redmine/scm/adapters/git_adapter_test.rb:288:in `test_revisions_disjointed_histories_revisio
ns'
    r:/Ruby193/lib/ruby/gems/1.9.1/gems/mocha-0.10.5/lib/mocha/integration/mini_test/version_230_to_262.rb:28:
in `run'
```

#### #79 - 2012-03-16 16:10 - Jeremy Bopp

It looks like the error listed is number 6 of some number of errors.  Are the other errors unrelated?  What version of Git was used in these tests?

**#80 - 2012-03-16 16:30 - Jeremy Bopp**

I have a theory about what might be causing the trouble on Windows.  See this line in #revisions:

```
io.puts(revisions.join("\n"))
```

This writes the revisions to the stdin of git-log, but it does it explicitly with Unix line endings between the revisions.  It probably needs to be "\r\n" on Windows.  Changing that line to this may fix the problem:

```
revisions.each {|r| puts(r)}
```

Note that this assumes that puts uses the correct line terminator when writing to a pipe on windows.  If it still uses just "\n", we'll have to try something else.

**#81 - 2012-03-16 17:25 - Toshi MARUYAMA**

*- Subject changed from Git: Too Long in fetching repositories after upgrade from 1.1 or new branch at first time to Git: Too long in fetching repositories after upgrade from 1.1 or new branch at first time*

*- Status changed from 7 to Closed*

*- % Done changed from 70 to 100*

*- Resolution set to Fixed*

Trunk r9144 is reasonable speed.

On bare repository.

```
$ git branch
* gh-new-0.6-stable
  gh-new-master

$ git log -n 1 gh-new-master
commit 8e76eac974675a6e34991bd54501e4284f88bfdd
Author: jplang <jplang@e93f8b46-1217-0410-a6f0-8f06a7374b81>
Date:   Mon Mar 5 11:03:26 2012 +0000

    Changed assertions to make them work with Rails2/3 ruby1.8/1.9 different behaviours.

    git-svn-id: svn://rubyforge.org/var/svn/redmine/trunk@9108 e93f8b46-1217-0410-a6f0-8f06a7374b81

$ git log gh-new-master  | grep ^commit | wc
   7788   15576  373824
```

On working area.

```
$ git checkout -b gh-new-master-01 16cb083a272cd0b
Switched to a new branch 'gh-new-master-01'

$ git log -n 1
commit 16cb083a272cd0b2ee3da954d7d8722453a86795
Author: jplang <jplang@e93f8b46-1217-0410-a6f0-8f06a7374b81>
Date:   Sun Mar 11 10:25:44 2012 +0000

    Additional tests for UsersController.

    git-svn-id: svn://rubyforge.org/var/svn/redmine/trunk@9231 e93f8b46-1217-0410-a6f0-8f06a7374b81

$ git log | grep ^commit | wc
   7880   15760  378240

$ git remote add localbare ../../git-bare-dir/redmine-00/

$ git push localbare gh-new-master-01 
Counting objects: 416, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (85/85), done.
Writing objects: 100% (351/351), 74.84 KiB, done.
Total 351 (delta 286), reused 325 (delta 264)
To ../../git-bare-dir/redmine-00/
 * [new branch]      gh-new-master-01 -> gh-new-master-01

$ time wget http://localhost:5000/projects/project6/repository

real    0m19.474s
```

```
user    0m0.001s
sys     0m0.006s
```

Note 76 patches do not work on Ruby 1.8.7.

```
/home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/activesupport-2.3.14/lib/active_support/dependencies.rb:184:in `re
quire': /REDMINE-1/hg-workdir/my-work/lib/redmine/scm/adapters/git_adapter.rb:231: syntax error, unexpected tS
YMBEG, expecting tAMPER (SyntaxError)
          scm_cmd(*cmd_args, :mode => "r+") do |io|
                              ^
/REDMINE-1/hg-workdir/my-work/lib/redmine/scm/adapters/git_adapter.rb:231: syntax error, unexpected ')', expec
ting kEND
          scm_cmd(*cmd_args, :mode => "r+") do |io|
                                           ^
/REDMINE-1/hg-workdir/my-work/lib/redmine/scm/adapters/git_adapter.rb:434: syntax error, unexpected tIDENTIFIE
R, expecting tAMPER
...se --verify --quiet|, revision) do |io|
                              ^
/REDMINE-1/hg-workdir/my-work/lib/redmine/scm/adapters/git_adapter.rb:440: syntax error, unexpected kRESCUE, e
xpecting kEND
            rescue ScmCommandAborted
                  ^
/REDMINE-1/hg-workdir/my-work/lib/redmine/scm/adapters/git_adapter.rb:448: syntax error, unexpected kEND, expe
cting $end
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/activesupport-2.3.14/lib/active_support/dependencies.rb:1
84:in `require'
    from /REDMINE-1/hg-workdir/my-work/app/models/repository/git.rb:19
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/activesupport-2.3.14/lib/active_support/dependencies.rb:1
84:in `require'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/activesupport-2.3.14/lib/active_support/dependencies.rb:1
84:in `require'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/activesupport-2.3.14/lib/active_support/dependencies.rb:2
91:in `require_or_load_without_engine_additions'
    from /REDMINE-1/hg-workdir/my-work/vendor/plugins/engines/lib/engines/rails_extensions/dependencies.rb:132
:in `require_or_load'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/activesupport-2.3.14/lib/active_support/dependencies.rb:2
50:in `depend_on'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/activesupport-2.3.14/lib/active_support/dependencies.rb:1
62:in `require_dependency'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/rails-2.3.14/lib/initializer.rb:414:in `load_application_
classes'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/rails-2.3.14/lib/initializer.rb:413:in `each'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/rails-2.3.14/lib/initializer.rb:413:in `load_application_
classes'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/rails-2.3.14/lib/initializer.rb:411:in `each'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/rails-2.3.14/lib/initializer.rb:411:in `load_application_
classes'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/rails-2.3.14/lib/initializer.rb:197:in `process'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/rails-2.3.14/lib/initializer.rb:113:in `send'
    from /home/xxxxxx/.rvm/gems/ruby-1.8.7-p357/gems/rails-2.3.14/lib/initializer.rb:113:in `run'
    from /REDMINE-1/hg-workdir/my-work/config/environment.rb:21
    from /REDMINE-1/hg-workdir/my-work/test/test_helper.rb:19:in `require'
    from /REDMINE-1/hg-workdir/my-work/test/test_helper.rb:19
    from test/unit/lib/redmine/scm/adapters/git_adapter_test.rb:3:in `require'
    from test/unit/lib/redmine/scm/adapters/git_adapter_test.rb:3
```

As I described at note-69,
Redmine 1.4.0 will be released on 2012-04-01.
So, I close this issue.
Please create a new issue when you finish to revise patches.

**#82 - 2012-03-19 16:12 - Jeremy Bopp**

As requested, issue #10470 has been created with updated and more thoroughly tested patches.  Please consider applying them for version 1.4.
Thank you.

## Files

| | | | |
|---|---|---|---|
| repo-setting.png | 36.6 KB | 2011-07-21 | Toshi MARUYAMA |
| Optimize-Git-operations-for-new-branches.patch | 15.2 KB | 2011-12-08 | Jeremy Bopp |
| Alias-the-revision-and-scmid-properties-for-Git-Revi.patch | 1.04 KB | 2011-12-08 | Jeremy Bopp |
| Optimize-Git-operations-for-new-branches.patch | 15.3 KB | 2011-12-16 | Jeremy Bopp |
| 0001-Performance-improvements-for-git-repo-parsing.patch | 5.57 KB | 2012-03-06 | Gergely Fábián |

| | | | |
|---|---|---|---|
| 0001-Git-repo-parsing-optimization-without-db-query-group.patch | 3.92 KB | 2012-03-08 | Gergely Fábián |
| 0001-Pass-revisions-to-git-log-via-stdin.patch | 7.79 KB | 2012-03-15 | Jeremy Bopp |
| 0002-Process-new-git-revisions-all-at-once-rather-than-pe.patch | 14.3 KB | 2012-03-15 | Jeremy Bopp |