Redmine - Patch #10470

Efficiently process new git revisions in a single batch

2012-03-16 21:16 - Jeremy Bopp

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Toshi MARUYAMA	**% Done:**	100%
Category:	SCM	**Estimated time:**	0.00 hour
Target version:	1.4.0		

Description

As noted in #8857, I am opening a new issue with patches that make processing new revisions with Git more efficient. I'm including the note that introduced the first pass at these patches below.

Here are 2 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 for #8857, 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.

These patches have been updated since they were submitted for #8857. They have been confirmed to work with the following Rubies:

- ruby 1.8.7 (2011-06-30 patchlevel 352) [x86_64-linux]
- ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-linux]
- ruby 1.9.3p0 (2011-10-30 revision 33570) [x86_64-linux]
- ruby 1.8.7 (2012-02-08 patchlevel 358) [i386-ming32]
- ruby 1.9.2p290 (2011-07-09) [i386-mingw32]
- ruby 1.9.3p0 (2011-10-30) [i386-mingw32]

Related issues:

Related to Redmine - Defect #8857: Git: Too long in fetching repositories aft...	**Closed**	**2011-07-20**

Associated revisions

Revision 9280 - 2012-03-28 12:54 - Toshi MARUYAMA

Ruby1.9: scm: use :set_encoding instead of "RUBY_VERSION < '1.9'" for IO.popen (#10470)

Revision 9281 - 2012-03-28 13:21 - Toshi MARUYAMA

scm: git: rename "scm_cmd" to "git_cmd" and not use variable argument (#10470)

Revision 9282 - 2012-03-28 17:40 - Toshi MARUYAMA

scm: git: use stdin instead of command line in "git log" (#10470)

Revision 9283 - 2012-03-28 19:18 - Toshi MARUYAMA

scm: git: process new git revisions all at once rather than per branch (#10470)

Revision 9284 - 2012-03-28 19:18 - Toshi MARUYAMA

scm: git: fix model source indents (#10470)

Revision 9287 - 2012-03-29 12:05 - Toshi MARUYAMA

scm: git: add the comment of the reason to scan database in fetching every time (#10470)

Revision 9288 - 2012-03-29 18:50 - Toshi MARUYAMA

scm: mercurial: transaction for each imported revisions (#10470)

Subversion transaction switched at r2563.
Subversion and Mercurial have a sequential revision number.
So, Mercurial can use the same logic with Subversion.

Revision 9311 - 2012-04-03 10:05 - Toshi MARUYAMA

scm: mercurial: git: save parents in creating changesets (#10470)

History

#1 - 2012-03-20 04:17 - llorbinshane aifseng

-

#2 - 2012-03-20 04:18 - uvernoiswayl billaa

-

#3 - 2012-03-20 09:20 - morlockmaria John

-

#4 - 2012-03-20 16:20 - Jeremy Bopp

There are less than 2 weeks left until the planned release date for version 1.4. Is there any chance that these patches will make it in or at least receive review?

#5 - 2012-03-22 01:03 - echelbill aifseng

-

#6 - 2012-03-22 01:07 - nieivesonstefa John

-

#7 - 2012-03-26 18:24 - Jeremy Bopp

One more try... Are these patches likely to receive any attention at all before version 1.4 is released?

#8 - 2012-03-28 13:51 - Alexey C.

+1 voting for that patch.

#9 - 2012-03-28 19:19 - Toshi MARUYAMA

- Target version set to 1.4.0

- % Done changed from 0 to 100

#10 - 2012-03-29 00:10 - Jeremy Bopp

First of all, thank you for incorporating my changes again. :-)

My only real concern with the way the patches were incorporated is the fact that all new revisions to process are loaded at once into an array before inserting them into the DB. This is in r9283. A repository with a sufficiently large number of commits could cause problems with memory consumption and the garbage collector as the array grows during the initial import of the repository. The original patches avoided this problem by passing a block to the revisions method that is then called once per revision.

The original solution may still run afoul of the garbage collector as revision objects are created and destroyed, but this is unavoidable in any case. However, it does avoid a potential for excessive memory growth overall. I think r9283 shouldn't be a problem for most repositories, but it is definitely vulnerable to initial imports of repositories with a very large number of commits.

#11 - 2012-03-30 12:51 - Toshi MARUYAMA

- Status changed from New to Closed

I cannot accept single transaction.
As I described r9288 comment, Subversion switched to transaction per revision at r2563.
Git also switched at trunk r3469 and 0.9-stable r3505.
These revisions are for 0.9.0 serious performance problems (#4547, #4716).

I cannot accept your "ignore_missing_revisions".
It calls many "git rev-parse --verify --quiet" if repository has many branches.
It causes serious performance problems.

- #8365
- #7047

As I described at #9472 note-23, history editing is rare case on shared repository.

As I described at source:trunk/app/models/repository/git.rb@9291#L176 ,
before pruning, "git log --not deleted_branch_head_revision" has no error.
And git 1.7.2.3 default pruning date is 2 weeks.
So, "git log" returns expected revisions.

Following is the example of **force push**.

```
$ git log -n 1 39e47ab99fd30c36 | cat
commit 39e47ab99fd30c3630994f67a8121447ff52daa4
Author: jplang <jplang@e93f8b46-1217-0410-a6f0-8f06a7374b81>
Date:   Tue Nov 29 21:04:58 2011 +0000

    Set version to 1.3.0.

    git-svn-id: svn://rubyforge.org/var/svn/redmine/trunk@7993 e93f8b46-1217-0410-a6f0-8f06a7374b81

$ git log -n 1 fc3b3c693470e506cb | cat
commit fc3b3c693470e506cbe0172a4e4514839f16004a
Author: jplang <jplang@e93f8b46-1217-0410-a6f0-8f06a7374b81>
Date:   Tue Nov 29 21:02:26 2011 +0000

    1.3-stable branch added.

    git-svn-id: svn://rubyforge.org/var/svn/redmine/branches/1.3-stable@7992 e93f8b46-1217-0410-a6f
0-8f06a7374b81
```

On working area.

```
$ git checkout -b testbranch-master 39e47ab99fd30c36
$ git checkout -b testbranch-1.3-stable fc3b3c693470e506cb
```

Force push.

```
$ git push localbare testbranch-master:testbranch

$ git push localbare testbranch-1.3-stable:testbranch
To ../../../git-baredir/redmine
 ! [rejected]        testbranch-1.3-stable -> testbranch (non-fast-forward)
error: failed to push some refs to '../../../git-baredir/redmine'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes (e.g. 'git pull') before pushing again.  See the
'Note about fast-forwards' section of 'git push --help' for details.

$ git push --force localbare testbranch-1.3-stable:testbranch
Counting objects: 1, done.
Writing objects: 100% (1/1), 255 bytes, done.
Total 1 (delta 0), reused 1 (delta 0)
Unpacking objects: 100% (1/1), done.
To ../../../git-baredir/redmine
 + 39e47ab...fc3b3c6 testbranch-1.3-stable -> testbranch (forced update)
```

"git log" has no error before "git gc".

```
$ ( echo ^39e47ab99fd30c36 ; echo fc3b3c693470e506cb ) | git --no-pager log --stdin
commit fc3b3c693470e506cbe0172a4e4514839f16004a
Author: jplang <jplang@e93f8b46-1217-0410-a6f0-8f06a7374b81>
Date:   Tue Nov 29 21:02:26 2011 +0000
```

```
    1.3-stable branch added.

    git-svn-id: svn://rubyforge.org/var/svn/redmine/branches/1.3-stable@7992 e93f8b46-1217-0410-a6f0-8f06a7374
b81
$ echo $?
0
```

"git log" has error after "git gc".

```
$ git gc --prune=1
$ ( echo ^39e47ab99fd30c36 ; echo fc3b3c693470e506cb ) | git --no-pager log --stdin
commit fc3b3c693470e506cbe0172a4e4514839f16004a
Author: jplang <jplang@e93f8b46-1217-0410-a6f0-8f06a7374b81>
Date:   Tue Nov 29 21:02:26 2011 +0000

    1.3-stable branch added.

    git-svn-id: svn://rubyforge.org/var/svn/redmine/branches/1.3-stable@7992 e93f8b46-1217-0410-a6f0-8f06a7374
b81

$ echo $?
0

$ git gc --prune=0
$ ( echo ^39e47ab99fd30c36 ; echo fc3b3c693470e506cb ) | git --no-pager log --stdin
fatal: bad revision '^39e47ab99fd30c36'

$ echo $?
128
```

#12 - 2012-04-02 16:02 - Jeremy Bopp

I'm not sure if the response was for my note of concern regarding the way the the revisions method was being used or not, but I don't believe it is. Please correct me if I'm wrong.

Toshi MARUYAMA wrote:

> I cannot accept single transaction.
> As I described r9288 comment, Subversion switched to transaction per revision at r2563.
> Git also switched at trunk r3469 and 0.9-stable r3505.
> These revisions are for 0.9.0 serious performance problems (#4547, #4716).

I looked over the revisions and issues you mentioned, but aside from it being a convenient way to handle the problem for Subversion and Mercurial, I can't agree that it was the only possible solution. Maybe I missed something.

In any case, the reason for the transaction over all newly processed revisions is to avoid the recorded heads falling out of sync with the revisions that were processed. Take note that the update of the list of heads is also managed in that single transaction. Without that transaction, concurrent attempts to process new revisions in the repository will race to update the last processed heads. Since the list of heads is set at the end of processing revisions, it's possible that the last seen heads list could be left with older heads if the first request to process the repository finishes last.

This is a minor issue since it's not likely that there will be very many reprocessed issues on a subsequent attempt to update the repository, but I want to make sure that it's not overlooked.

> I cannot accept your "ignore_missing_revisions".
> It calls many "git rev-parse --verify --quiet" if repository has many branches.
> It causes serious performance problems.
>
> - #8365
> - #7047

I agree that it's not a great solution. I implemented that when one of the existing tests failed while attempting to emulate a history rewrite. It purposely set a non-existent revision in the seen heads list before triggering the test repository to be processed. That phoney revision caused git-log to fail in much the same way a pruned head revision would.

There is a new option coming for the git-log command that would integrate this behavior and avoid the need to prune the last seen heads list explicitly like that. Take a look at the --ignore-missing option here. Sadly, I'm not sure what version of git actually has that, but the version I have (1.7.5.4) doesn't. Perhaps this functionality can wait until git versions with that feature are more widespread.

> As I described at #9472 note-23, history editing is rare case on shared repository.

The problem here could happen simply by **deleting** a branch that hasn't been merged into any other and waiting a while before letting Redmine

process the repository again. While this is a bit of a corner case, Redmine will attempt to feed the last known revision for the deleted branch to git-log. If that revision has been pruned because the branch that referenced it was removed more than 2 weeks ago (under the current default setting for git), git-log will fail, and no new revisions would be processed.

The only obvious solution then would be to reprocess the repository from scratch, and that can take time and has risks for wrecking issue states if people aren't careful to disable fixing keyword handling before beginning.

Granted, this should be uncommon, but it's still something that concerns me. If the overhead of cleaning up the old heads list is as bad as you imply, perhaps a reasonable workaround would be to add the head cleanup logic as a separate function that could be used if the repository wound up in this state. The easiest thing would probably be to include a script whose sole purpose is to eliminate invalid heads from the last seen heads list. It may never get run, but it's a pretty easy one to write just in case.

#13 - 2012-04-02 17:32 - Toshi MARUYAMA

About memory consumption,
I agree Gergely's #8857 note-50 .

> So the 'all_revisions' array for a repository with 30000 commits would consume 1.171 MiB.
> I guess, it's not so much.

Changeset table revision column is unique.

- source:tags/1.3.2/db/migrate/034_create_changesets.rb#L10
- source:tags/1.3.2/db/migrate/091_change_changesets_revision_to_string.rb

Subversion and Mercurial revision number are saved in revision column.
and these are unique on its repository.

Git revision hash value is saved in revision column and it is unique.

In case of multiple fetching processes in transaction per revision,
if revision exists on database, saving fails.
So, "changeset.new_record?" is true.
source:tags/1.3.2/app/models/repository/subversion.rb#L75

In case of single transaction, many memory consumption in **database layer**.
And all data lose if database stops in the middle of transaction.

#14 - 2012-04-02 23:53 - Jeremy Bopp

Toshi MARUYAMA wrote:

> About memory consumption,
> I agree Gergely's #8857 note-50 .
>
> > So the 'all_revisions' array for a repository with 30000 commits would consume 1.171 MiB.
> > I guess, it's not so much.

Why be unnecessarily wasteful when the functionality to avoid it already exists? Also consider that building an array of revisions one at a time will also trigger the growth of the array structure itself. It's probably not a very significant performance penalty most of the time, but again, why be wasteful here?

In other words, I'm not seeing any argument **for** loading things all at once except for basically "It's already done that way." :-)

> Changeset table revision column is unique.
>
> - source:tags/1.3.2/db/migrate/034_create_changesets.rb#L10
> - source:tags/1.3.2/db/migrate/091_change_changesets_revision_to_string.rb
>
> Subversion and Mercurial revision number are saved in revision column.
> and these are unique on its repository.
>
> Git revision hash value is saved in revision column and it is unique.
>
> In case of multiple fetching processes in transaction per revision,
> if revision exists on database, saving fails.
> So, "changeset.new_record?" is true.
> source:tags/1.3.2/app/models/repository/subversion.rb#L75
>
> In case of single transaction, many memory consumption in **database layer**.

I believe this consumption would actually live in the DB server itself, and for non-trivial DB implementations, this should be handled efficiently.

> And all data lose if database stops in the middle of transaction.

That is as it should be in my opinion. As I mentioned earlier, using a transaction ensures that the list of last processed heads corresponds to the list of processed revisions. Since there isn't a severe penalty that I can think of should the heads list fall out of sync though, it's reasonable to be a little sloppy here if the performance benefits of avoiding a transaction are significant. However, I doubt the transaction would routinely last long enough to make a measurable difference either way. If someone pushes 10,000 commits all at once to the repository, you might have a problem.

To be honest, it would probably be far better to avoid the DB entirely for Git repositories and always probe them on demand. If I recall correctly, that solution was rejected or perhaps simply sidestepped in earlier issues though. Too bad.

#15 - 2012-04-03 00:42 - Toshi MARUYAMA

Jeremy Bopp wrote:

> Toshi MARUYAMA wrote:
>
> > About memory consumption,
> > I agree Gergely's #8857 note-50 .
> >
> > > So the 'all_revisions' array for a repository with 30000 commits would consume 1.171 MiB.
> > > I guess, it's not so much.
>
> Why be unnecessarily wasteful when the functionality to avoid it already exists? Also consider that building an array of revisions one at a time will also trigger the growth of the array structure itself. It's probably not a very significant performance penalty most of the time, but again, why be wasteful here?

As I described at source:tags/1.3.2/app/models/repository/git.rb#L125 ,
Git 1.7.3.4 does not support --reverse with -n or --skip.

Subversion reads 200 revisions.
source:tags/1.3.2/app/models/repository/subversion.rb#L62
Mercurial reads 100 revisions.
source:app/models/repository/mercurial.rb@9288#L141

#16 - 2012-04-03 05:30 - Jeremy Bopp

I apologize for this very lengthy note in advance, but I feel the need to be thorough here because we seem to be miscommunicating somewhere. Please be patient and digest everything. :-)

Toshi MARUYAMA wrote:

> Jeremy Bopp wrote:
>
> > Toshi MARUYAMA wrote:
> >
> > > About memory consumption,
> > > I agree Gergely's #8857 note-50 .
> > >
> > > > So the 'all_revisions' array for a repository with 30000 commits would consume 1.171 MiB.
> > > > I guess, it's not so much.
> >
> > Why be unnecessarily wasteful when the functionality to avoid it already exists? Also consider that building an array of revisions one at a time will also trigger the growth of the array structure itself. It's probably not a very significant performance penalty most of the time, but again, why be wasteful here?
>
> As I described at source:tags/1.3.2/app/models/repository/git.rb#L125 ,
> Git 1.7.3.4 does not support --reverse with -n or --skip.

Looking at the current trunk code (source:trunk/app/models/repository/git.rb@9310#L190), it appears that the real reason for handling the revisions as a single array is the explicit filtering of the returned list of revisions based on revisions already in the DB, 100 items at a time. The combination of options you mention is irrelevant with what you have already merged from my patches. The -n and --skip options are not used and would not be helpful even if they *could* be used.

The current code actually has at least 3 problems:

1. It attempts to filter revisions that are almost guaranteed to be new.
2. It contains a race condition.
3. It most likely incurs unnecessary overhead for many little transactions.

Filtering Revisions

Keep in mind that contrary to the comments above the referenced code, the code will not help at all if there are "tens of thousands" or really any number of **new** revisions to process. Due to the functionality you **did** integrate from my patches, git itself eliminates the vast majority of already processed revisions based on the last seen heads. The code cannot filter out anything as a result because the returned revisions are new by definition and must **all** be processed. All the code will do is run many queries that return empty sets, thereby causing the array of revisions to be needlessly duplicated 100 elements at a time only to then be processed in its entirety. In addition the code still inserts the revisions into the DB one at a time, so there is no batch processing as far as the DB is concerned, just tens of thousands individual insert operations with their associated round trips to and from the DB.

As noted in the comments for the referenced code, the re-addition of a formerly deleted branch could cause the reprocessing of revisions that are unique to that branch. However, that should be a rare event since most branches would only be deleted after merging them into other branches in the first place or when their unique revisions are truly undesirable. In addition, most branches only pull in a relative handful of revisions that are not already reachable from other heads in the repository, so even if the revisions unique to a branch are re-processed, this should only be a tiny amount of additional work for the DB to reject the insert attempts. Filtering the revisions in the application using a DB lookup probably eats just as much time in the end.

This logic complicates the code with very little real benefit. I believe it is a relic of the method Gergely used in his patches from #8857. His patches would re-process virtually the entire repository for each new branch, if I recall correctly, and therefore needed to filter out the bulk of revisions for any new branch. It's the same problem my original patches (and these new ones) were intended to avoid from the outset: let git filter out already processed revisions even for new branches.

Race Condition

Imagine that 2 threads are processing the same set of new heads. Both fetch the same set of new revisions from the revisions method. Now, both start filtering their revision lists 100 revisions at a time; however, the first thread finishes first and starts inserting new revisions while the second thread finishes filtering the last few revisions in its list. The first thread successfully inserts each new revision into the DB. The second thread eventually begins attempting to insert revisions, but each of them has already been inserted by the first thread. They weren't filtered at the time because the first thread hadn't inserted them yet. This won't cause any problems since the attempts to insert will be rejected, but it is a wasted bit of processing that would have been completely prevented if a transaction around the whole update process had been used.

Transaction Overhead

For any database system that supports real transactions, there is overhead associated with setting them up and tearing them down. The current code creates a transaction for each individual revision to be inserted into the DB. That overhead is going to add up when processing hundreds of revisions or more. It would be better to simply run it all within a single transaction.

Summary

Processing the revisions using a block as I did in my patches will not cause undue load on the DB since the revisions will most likely be new and need to be inserted. This should also decrease the load on the application since needless filtering is avoided. Using a transaction around the entire set of processing should eliminate the race condition, reduce the overhead of setting up and tearing down a transaction for each inserted revision, and only hold a lock on the DB long enough to process the **new** revisions due to the more optimized use of git. The only times I can think that the single transaction should be problematic in the real world are:

1. The initial import of a new repository
   - Solution: Do that during server downtime.
2. The DB is too flaky to process everything in a single transaction
   - Solution: Replace that DB before it eats your parents! ;-)

#17 - 2012-04-03 10:16 - Toshi MARUYAMA

Jeremy Bopp wrote:

> 1. The initial import of a new repository
> - Solution: Do that during server downtime.

Yes.
Otherwise:

- not use "Fetch commits automatically" setting
- not use hook and use cron job

> 1. The DB is too flaky to process everything in a single transaction
> - Solution: Replace that DB before it eats your parents! ;-)

I committed r9311.
"git log --reverse" and "hg log -r lesser:greater" return parents first.

#18 - 2012-04-03 16:41 - Jeremy Bopp

Toshi MARUYAMA wrote:

> I committed r9311.
> "git log --reverse" and "hg log -r lesser:greater" return parents first.

The changes introduced by r9311 do not appear to be related to your comment here. Did you intend to refer to a different revision?

#19 - 2012-04-03 17:26 - Toshi MARUYAMA

Jeremy Bopp wrote:

> Toshi MARUYAMA wrote:
>
> > I committed r9311.
> > "git log --reverse" and "hg log -r lesser:greater" return parents first.
>
> The changes introduced by r9311 do not appear to be related to your comment here. Did you intend to refer to a different revision?

No.

#20 - 2012-04-03 17:35 - Jeremy Bopp

Toshi MARUYAMA wrote:

> Jeremy Bopp wrote:
>
> > Toshi MARUYAMA wrote:
> >
> > > I committed r9311.
> > > "git log --reverse" and "hg log -r lesser:greater" return parents first.
> >
> > The changes introduced by r9311 do not appear to be related to your comment here. Did you intend to refer to a different revision?
>
> No.

Then I'm confused. Sorry.

r9311 is a good change but unrelated to anything I've been trying to discuss with you up until this point. Perhaps your comment was intended to explain that the change introduced by r9311 is safe *because* "git log --reverse" ensures that the ancestor revisions are always processed before descendent revisions?

#21 - 2012-04-03 17:39 - Toshi MARUYAMA

Jeremy Bopp wrote:

> Toshi MARUYAMA wrote:
>
> > Jeremy Bopp wrote:
> >
> > > Toshi MARUYAMA wrote:
> > >
> > > > I committed r9311.
> > > > "git log --reverse" and "hg log -r lesser:greater" return parents first.
> > >
> > > The changes introduced by r9311 do not appear to be related to your comment here. Did you intend to refer to a different revision?
> >
> > No.
>
> Then I'm confused. Sorry.
>
> r9311 is a good change but unrelated to anything I've been trying to discuss with you up until this point. Perhaps your comment was intended to explain that the change introduced by r9311 is safe *because* "git log --reverse" ensures that the ancestor revisions are always processed before descendent revisions?

Yes.

Files

0001-Pass-revisions-to-git-log-via-stdin.patch	10.4 KB	2012-03-16	Jeremy Bopp
0002-Process-new-git-revisions-all-at-once-rather-than-pe.patch	14.3 KB	2012-03-16	Jeremy Bopp