Redmine - Defect #4773
Redmine+Git+PostgresSQL 8.4 fails with linux kernel tree (encoding)

2010-02-09 03:27 - Oren Laadan

Status:	Closed	**Start date:**	2010-02-09
Priority:	Normal	**Due date:**	
Assignee:	Jean-Philippe Lang	**% Done:**	0%
Category:	Database	**Estimated time:**	0.00 hour
Target version:	0.9.3		
Resolution:	Fixed	**Affected version:**	0.8.6

Description

Hi,

I'm running into an issue similar to Issue #1663.

I also hit it before - see http://www.redmine.org/boards/2/topics/9743.
Now I tried again with recent Redmine, and still no luck.

I want to use redmine for development on the linux kernel git tree.
(this takes the longest time ever... see Issue #4547).

I started with the detault setting of UTF8 for commit messages, but
the import eventually failed because the git tree had a commit with
encoding ISO-8859-1. So I started from scratch, with the ISO-8859-1
settings instead, but this too eventually failed because the git tree
had a commit with a UTF8 encoding.

Catch-22 ?

Is there a quick-and-dirty way to make redmine convert offending
commit messages (e.g. that are not in UTF8 or compliant form) on the
fly and avoid this annoying pitfall ?

Many thanks,

Oren.

About your application's environment
Ruby version 1.8.7 (i486-linux)
RubyGems version 1.3.5
Rack version 1.0
Rails version 2.3.5
Active Record version 2.3.5
Active Resource version 2.3.5
Action Mailer version 2.3.5
Active Support version 2.3.5
Application root /srv/redmine.git
Environment production
Database adapter postgresql
Database schema version 20091227112908

About your Redmine plugins
Redmine Wiki Extensions plugin 0.2.0

Related issues:

Related to Redmine - Defect # 4547: git: Very high CPU usage for a long time ...	**Closed**	**2010-01-11**
Related to Redmine - Defect # 7047: Git adapter very slow when a commit modif...	**New**	**2010-12-04**
Related to Redmine - Feature # 3396: Git: use --encoding=UTF-8 in "git log"	**Closed**	**2009-05-20**
Related to Redmine - Defect # 2664: Mercurial: Repository path encoding of no...	**Closed**	**2009-02-04**

Associated revisions

Revision 3466 - 2010-02-20 12:24 - Jean-Philippe Lang

Remove invalid utf8 sequences from commit comments and author name (#4773).

Revision 4926 - 2011-02-22 14:39 - Toshi MARUYAMA

scm: for log in Ruby 1.9, replace invalid UTF-8 to '?' instead of removing.

Refer r3466 #4773.

Revision 5644 - 2011-05-05 01:38 - Toshi MARUYAMA

scm: git: add comments of revision order in fetch_changesets().

Related issues.
#5357, #6013, #7146, #4773, #4547, #1406, #3449, #3567.

History

#1 - 2010-02-09 19:57 - Jean-Philippe Lang

Is there a proper way to make git output commit messages in a given encoding?

#2 - 2010-02-09 23:41 - Oren Laadan

The manpage of 'git log' says:

> --encoding[=<encoding>]
> The commit objects record the encoding used for the log message in their encoding header; this option can be used to tell the command to re-code the commit log message in the encoding preferred by the user. For non plumbing commands this defaults to UTF-8.

However, the discussion adds:

> git-log, git-show, git-blame and friends look at the encoding header of a commit object, and try to re-code the log message into UTF-8 unless otherwise specified....

IOW, this only works if the commit message has a header that indicated its encoding. I manually checked the commit in question and I don't think that it has one (commit 4d900f9df5f0569c2dc536701e2c11b6d50ebebf of the linux kernel git repository).

So while you probably want to use "--encoding=XXXX" in the git command as per the selected DB encoding, it's still not a panacea.

Oren

#3 - 2010-02-09 23:50 - Oren Laadan

How about using piping the output through iconv(1) like:

```
git log ....  | iconv -f UTF8 -c
```

Where (from the manpage of iconv):

```
-f     Convert characters from encoding.
-c     Omit invalid characters from output.
```

And replace UTF8 by the selected encoding, if needed.

Or use the ruby's iconv() function, similarly.

#4 - 2010-02-10 22:59 - Jean-Philippe Lang

Can you try this patch:

```
Index: app/models/changeset.rb
===================================================================
--- app/models/changeset.rb    (revision 3404)
+++ app/models/changeset.rb    (working copy)
@@ -180,11 +180,12 @@
     encoding = Setting.commit_logs_encoding.to_s.strip
     unless encoding.blank? || encoding == 'UTF-8'
       begin
-        return Iconv.conv('UTF-8', encoding, str)
+        str = Iconv.conv('UTF-8', encoding, str)
       rescue Iconv::Failure
         # do nothing here
       end
     end
-    str
+    # removes invalid UTF8 sequences
```

```
+    Iconv.conv('UTF-8//IGNORE', 'UTF-8', str + '  ')[0..-3]
   end
 end
```

#5 - 2010-02-11 16:58 - Oren Laadan

Good news and bad news:

The good news is that the patch works, and the repository import
proceeded beyond the point it had failed before.

The bad news is that it eventually failed anyways -- and this time,
it appears, because the committer (user) name was not in UTF8.

Here is the error message from the log:

> ActiveRecord::StatementInvalid (PGError: ERROR: invalid byte sequence for encoding "UTF8": 0xf66e6967
> HINT: This error can also happen if the byte sequence does not match the encoding expected by the server, which is controlled by "client_encoding".
> : SELECT "changesets"."id" AS t0_r0, "changesets"."repository_id" AS t0_r1, "changesets"."revision" AS t0_r2, "changesets"."committer" AS t0_r3, "changesets"."committed_on" AS t0_r4, "changesets"."comments" AS t0_r5, "changesets"."commit_date" AS t0_r6, "changesets"."scmid" AS t0_r7, "changesets"."user_id" AS t0_r8, "users"."id" AS t1_r0, "users"."login" AS t1_r1, "users"."hashed_password" AS t1_r2, "users"."firstname" AS t1_r3, "users"."lastname" AS t1_r4, "users"."mail" AS t1_r5, "users"."mail_notification" AS t1_r6, "users"."admin" AS t1_r7, "users"."status" AS t1_r8, "users"."last_login_on" AS t1_r9, "users"."language" AS t1_r10, "users"."auth_source_id" AS t1_r11, "users"."created_on" AS t1_r12, "users"."updated_on" AS t1_r13, "users"."type" AS t1_r14, "users"."identity_url" AS t1_r15 FROM "changesets" LEFT OUTER JOIN "users" ON "users".id = "changesets".user_id AND ("users"."type" = 'User' OR "users"."type" = 'AnonymousUser') WHERE ("changesets".repository_id = 14 AND ("changesets"."committer" = E'Uwe Kleine-K<F6>nig <Uwe.Kleine-Koenig@digi.com>')) ORDER BY changesets.committed_on DESC, changesets.id DESC LIMIT 1):

Specifically, see the WHERE clause.

This is probably due to commit bc3c26fe65ecaa3fa96844219a9070a3e079697a in the linux
kernel git repository.

I suspect you need to do a similar trick with the 'committer' field ?

(unfortunately I have near zero knowledge in ruby...)

Thanks !

#6 - 2010-02-13 10:54 - Jean-Philippe Lang

Here is the additional patch:

```
Index: app/models/changeset.rb
===================================================================
--- app/models/changeset.rb    (revision 3420)
+++ app/models/changeset.rb    (working copy)
```

```
@@ -57,6 +57,10 @@
     super
   end

+  def committer=(arg)
+    write_attribute(:committer, self.class.to_utf8(arg.to_s))
+  end
+
   def project
     repository.project
   end
```

#7 - 2010-02-16 14:41 - Oren Laadan

Ok, that indeed works, great !
Many thanks for the helpful support.

On a related topic - related to issue #4547: all in all it took a few hours (!!!) to import the entire git repository of the linux kernel. This is after applying a patch to not use "--find-copies-harder" -- if not removed, the import took too much memory and time.

The rate commits import was around 250/min at the beginning, and then dropped to ~60/min at the end.

This cost is a very high, but is acceptable as a one time cost that is needed to setup the repository. Unfortunately, it appears that it recurs every time the git repository is updated (after each new commit). This severely limits the usability of redmine.

One thought I had, is to have a special option (for project admin) to request a full scan of the git repository. This is a long operation and will be very infrequent.
The usual option would be (e.g. when clicking on 'repository') to assume that the developers are sane, and only add commits to the public tree. I assume that in this case, a full rescan of the tree isn't required.
I'll be more than happy to assist with testing and deployment if it can be useful for you.

Any word on attempts to solve this ?

Many thanks,

Oren.

#8 - 2010-02-16 15:22 - Jean-Philippe Lang

Oren Laadan wrote:

> *Ok, that indeed works, great !*

> *Many thanks for the helpful support.*

Thanks for the feedback. I'll commit these pacthes.

> *This cost is a very high, but is acceptable as a one time cost that*
> *is needed to setup the repository. Unfortunately, it appears that it*
> *recurs every time the git repository is updated (after each new commit).*
> *This severely limits the usability of redmine.*
>
> *One thought I had, is to have a special option (for project admin) to*
> *request a full scan of the git repository. This is a long operation and*
> *will be very infrequent.*
> *The usual option would be (e.g. when clicking on 'repository') to assume*
> *that the developers are sane, and only add commits to the public tree.*
> *I assume that in this case, a full rescan of the tree isn't required.*
> *I'll be more than happy to assist with testing and deployment if it can*
> *be useful for you.*

The full re-scan behaviour was introduced in r2840 but several users reported this as a defect (#4547, #4716).
A fix was committed in r3394 (trunk), so that only commits from the last 7 days are scanned.

Can you give it a try?

#9 - 2010-02-19 04:20 - Oren Laadan

Oh ... excellent !

I tried to "rescan" after adding about 80 commits, and it took
about 2-3 minutes. During most of this time the postgres process
was chewing 100% cpu, by the way. However, this is now doable.

So this is definitely step in the right direction and makes the
system usable again.

Is it possible to configure that choice of 7 days ?

Is it possible to request a full scan ? (sure, once in a blue
moon, but still).

And also a related question: will it break anything if I disable
the "respository" module entirely, and instead use gitweb (the git
web browsing) and add a tab in the project that embeds the gitweb
page with an iframe ?

More specifically, will I lose any important (or less important)
features of redmine by not choosing the "repository" module for
the project altogether ?

The advantage of using gitweb instead is that it's designed to
provide fast and effective access to the repository, it is the

standard way for git repositories, and it's already familiar for
many users.

Thanks !

#10 - 2010-02-20 12:36 - Jean-Philippe Lang

- Status changed from New to Resolved
- Target version set to 0.9.3
- Resolution set to Fixed

The patches are committed in r3466 (trunk).

Oren Laadan wrote:

> *Oh ... excellent !*
>
> *I tried to "rescan" after adding about 80 commits, and it took about 2-3 minutes. During most of this time the postgres process was chewing 100% cpu, by the way. However, this is now doable.*
>
> *So this is definitely step in the right direction and makes the system usable again.*

Well, 2 minutes for 80 commits is still a bit long.
I'll see if it can be improved.

> *Is it possible to configure that choice of 7 days ?*

For now, it's not. But that would be pretty easy to add an option for that.

> *Is it possible to request a full scan ? (sure, once in a blue moon, but still).*

No. The only way for now is to remove the repository from Redmine and reload it...
The way it was done before r3394 is not a reasonable solution for large repositories (all the changeset from the Redmine database and all the git commits were loaded in memory).

> *And also a related question: will it break anything if I disable the "respository" module entirely, and instead use gitweb (the git web browsing) and add a tab in the project that embeds the gitweb page with an iframe ?*
>
> *More specifically, will I lose any important (or less important) features of redmine by not choosing the "repository" module for the project altogether ?*

Feature that are important to me with the repository module are:

1. ability to link commits with issues (viewing the associated commits from the issue view is a real plus)

2. commits comments search
3. commits displayed in the activity view

#11 - 2010-02-20 22:27 - Oren Laadan

> *I tried to "rescan" after adding about 80 commits, and it took about 2-3 minutes. During most of this time the postgres process was chewing 100% cpu, by the way. However, this is now doable.*
>
> *Well, 2 minutes for 80 commits is still a bit long.*
> *I'll see if it can be improved.*

I thought it would be useful so I timed how long it takes to rescan after adding a single commit. The total time was ~65 seconds, with the following breakdown:

1. First, postgres ran for ~50 seconds (postgres 85% cpu, ruby 15%)
2. Then, git ran for ~10 seconds (nearly 90% cpu)
3. Finally, postgress and ruby ran for ~5 seconds (mix cpu usage)

In addition, access to the repository without a rescan - e.g. to view the repository or to display recent activity - takes about 6 seconds, roughly split half and half between git work and ruby work. This is pretty long response time :(

> *Is it possible to configure that choice of 7 days ?*
>
> *For now, it's not. But that would be pretty easy to add an option for that.*

This is very important because a common workflow with git is to write a patch, then post it for review and only much later push to the main repository. Therefore, often the patch is already more than a week old when it gets to be an official changeset.

> *Is it possible to request a full scan ? (sure, once in a blue moon, but still).*
>
> *No. The only way for now is to remove the repository from Redmine and reload it...*
> *The way it was done before r3394 is not a reasonable solution for large repositories (all the changeset from the Redmine database and all the git commits were loaded in memory).*

This is problematic because limiting the rescan to a window of time to the past means that eventually there will be a commit (or more) that won't be scanned.

Perhaps there is a better approach: is it possible to scan and *add* everything that has changed since last scan, without worrying about existing changesets that may have changed ?

In particular, public repositories tend to grow "forward" and never modfy their own history. Changing one's history is considered very bad practice, so redmine can rely on such behavior (and declare that as a requirement).

Given this assumption, the rescan algorithm can be something like this:

- Compare the current repository state with the previous one, and
- Remove from the DB branches that were deleted (no longer exist)
- Add to the DB branches that are new (did not exist before)
- For each branch, add to the DB the changesets since the last HEAD

This solution does not depend on how long ago changesets appeared in the repository, but only on what was added since last time. It rids the need for the costly pass that checks all changesets against the redmine's DB.

If a manager/developer really wants to change the history of a branch, she could delete the branch, rescan, then re-add the branch and then rescan again. I suspect this will be very rare.

> *And also a related question: will it break anything if I disable the "respository" module entirely, and instead use gitweb (the git web browsing) and add a tab in the project that embeds the gitweb page with an iframe ?*
>
> *Feature that are important to me with the repository module are:*
>
> *1. ability to link commits with issues (viewing the associated commits from the issue view is a real plus)*
> *2. commits comments search*
> *3. commits displayed in the activity view*

I'm trying to figure out how hard it would be to achieve the same thing by leveraging gitweb instead. For example how I set it up, take a look at http://www.linux-cr.org/redmine/tab/show/user-cr. (This is a smaller repository, the linux-kernel based project is still not public, but if you register I can set up access for you there).

1. How difficult is it to allow to link a commit with an issue by way of a gitweb link instead of the local redmine's DB ? it will still be a click away, but will go to the gitweb tab ? For example, the wiki formatting could be replace "commit:c6f4d0fd" with "gitweb:c6f4d0fd"

2. This is also possible with gitweb, and is pretty efficient and neat.

3. Probably more involved, but essentially one could replace the DB query (that I assume is what you use to generate the report) with a combination of DB query and git query, and generate the changeset report on the fly from git (instead of from redmine's DB). Is that possible ?

#12 - 2010-02-28 10:56 - Jean-Philippe Lang

- Status changed from Resolved to Closed

The fix concerning invalid utf8 sequences is merged in 0.9-stable in r3501 so I close this ticket.

> *Perhaps there is a better approach: is it possible to scan and add everything that has changed since last scan, without worrying about existing changesets that may have changed ?*

True. How do you know about what was added since last visit?

> *I'm trying to figure out how hard it would be to achieve the same thing by leveraging gitweb instead.*

Actually, what is really slow when browsing a git repository is to get information about last change on each file using git log -n 1. When removing this, browsing is pretty fast. Maybe we should disable this.

#13 - 2010-03-02 03:24 - Oren Laadan

> *Perhaps there is a better approach: is it possible to scan and add everything that has changed since last scan, without worrying about existing changesets that may have changed ?*

> *True. How do you know about what was added since last visit?*

I'm not familiar with how redmine tracks the repositories. It seem that it stores some metadata of all the commits in the db, but otherwise calls the repository backend (git) to get the log and etc. Specifically, it does not record any state related branches and the HEAD of each branch.

So I'd suggest a new table that will record the "last known state" of the repository, with at least two columns: "branch" for branch name, and "commit" to identify the branch HEAD. (Maybe also a "date" to record when it was last updated). Whenever the database is rescanned, the table will be updated, so you will know which branches were added, removed, and updated. Then you ask git to only report (int each branch) what comes after the HEAD of each modified branch.

Does this make sense ?

> *I'm trying to figure out how hard it would be to achieve the same thing by leveraging gitweb instead.*
>
> *Actually, what is really slow when browsing a git repository is to get information about last change on each file using git log -n 1. When removing this, browsing is pretty fast. Maybe we should disable this.*

Do you have a patch that I can try that here ?

Thanks !

#14 - 2010-03-02 04:18 - Toshi MARUYAMA

Oren Laadan wrote:

> *So I'd suggest a new table that will record the "last known state" of the repository, with at least two columns: "branch" for branch name, and "commit" to identify the branch HEAD.*

Mercurial has same problem and try to resolve at #4455 - Mercurial overhaul.
I agree this suggestion to add new record at table.

#15 - 2010-03-02 05:30 - Toshi MARUYAMA

- File git-fast-browse.patch added

Oren Laadan wrote:

> *Actually, what is really slow when browsing a git repository is to get information about last change on each file using git log -n 1. When removing this, browsing is pretty fast. Maybe we should disable this.*
>
> *Do you have a patch that I can try that here ?*
>
> *Thanks !*

Try this patch.

#16 - 2010-03-04 03:01 - Oren Laadan

> > *Actually, what is really slow when browsing a git repository is to get information about last change on each file using git log -n 1. When removing this, browsing is pretty fast. Maybe we should disable this.*
>
> *Do you have a patch that I can try that here ?*
>
> *Try this patch.*

Yes, I can confirm that this is much faster.

#17 - 2010-03-04 03:03 - Oren Laadan

> > *So I'd suggest a new table that will record the "last known state" of the repository, with at least two columns: "branch" for branch name, and "commit" to identify the branch HEAD.*
>
> *Mercurial has same problem and try to resolve at #4455 - Mercurial overhaul.*
>
> *I agree this suggestion to add new record at table.*

Is there work in progress ? I'll be more than happy to testing and evaluation.

#18 - 2010-03-04 03:13 - Toshi MARUYAMA

Oren Laadan wrote:

> *Is there work in progress ? I'll be more than happy to testing and evaluation.*

This is only for Mercurial.
http://github.com/marutosi/redmine/commits/hg-overhaul

I have a plan to add parent1, parent2, named_branch field on changesets table, and add new ScmHeads table.

#19 - 2010-09-24 00:54 - Toshi MARUYAMA

Toshi MARUYAMA wrote:

> *I have a plan to add parent1, parent2, named_branch field on changesets table, and add new ScmHeads table.*

Mercurial can have only two parents, but Git can have multi parents.

Files

git-fast-browse.patch	678 Bytes	2010-03-02	Toshi MARUYAMA