Ruby master - Feature #5427

Not complex patch to improve `require` time (load.c)

10/09/2011 05:49 PM - funny_falcon (Yura Sokolov)

Status:	Closed
Priority:	Normal
Assignee:	nobu (Nobuyoshi Nakada)
Target version:	

Description

Currently loaded_features are unsorted, so that rb_feature_p ought to iterate over all loaded_features to figure: is requested feature loaded?

After this patch loaded_features is kept sorted by basename without extension (/usr/lib/ruby/asdf.rb => asdf). When rb_feature_p start its check, it goes straight to the first item with matching basename (using binary search) and stops after last.

Methods $LOADED_FEATURES.<< and $LOADED_FEATURES.push are overriden to keep sort order. push accepts only 1 parameter, but it seems that no one pass more to it.

Currently I choose to consider characters of basename in right to left order, but it could be changed, I think.

https://gist.github.com/1272991
https://github.com/ruby/ruby/pull/51

Related issues:

Has duplicate Ruby master - Feature #7387: Keep LOADED_FEATURES sorted by fil...	**Closed**	**11/19/2012**

History

#1 - 10/09/2011 05:51 PM - funny_falcon (Yura Sokolov)

Is there any chance for it or something like it to be included in 1.9.3 release?

#2 - 10/10/2011 05:23 AM - naruse (Yui NARUSE)

It's too late for 1.9.3.
Its feature is frozen on July 10.
http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/37865

#3 - 10/12/2011 01:02 AM - funny_falcon (Yura Sokolov)

11.10.2011 06:42, Urabe Shyouhei пишет:

> On 10/10/2011 07:51 PM, Yura Sokolov wrote:
>
> > Anyway, is idea looks good?
>
> The approach is interesting. Do you have any benchmarks for how fast is it?
>
> With my application
> Before patch:

```
RAILS_ENV=production rails runner 'puts $LOADED_FEATURES.size'
985

real    0m6.089s
user    0m5.644s
sys 0m0.308s
```

After patch:

```
RAILS_ENV=production rails runner 'puts $LOADED_FEATURES.size'
985

real    0m5.059s
user    0m4.552s
sys 0m0.336s
```

#4 - 10/12/2011 03:51 AM - Eregon (Benoit Daloze)

On 11 October 2011 04:42, Urabe Shyouhei wrote:

> On 10/10/2011 07:51 PM, Yura Sokolov wrote:
>
> > Anyway, is idea looks good?
>
> The approach is interesting. Do you have any benchmarks for how fast is it?

I was also curious how it performed so I made a benchmark.

Please have a look at https://gist.github.com/1278881

#5 - 10/13/2011 01:15 AM - jonforums (Jon Forums)

Patch applies cleanly, builds on my Win7 32bit using TDM-GCC 4.6.1 at ruby 1.9.3dev (2011-10-11 revision 33457) [i386-mingw32] and passes all make test-all TESTS='openssl fiddle psych' && make test

Planning to benchmark and test some more, but simple things like gem --version, gem outdated, gem list, and gem --version seem to be noticeably snappier in my Win7 shell.

#6 - 10/13/2011 04:23 AM - yeban (Anurag Priyam)

On Mon, Oct 10, 2011 at 1:53 AM, Yui NARUSE naruse@airemix.jp wrote:
[...]

> It's too late for 1.9.3.
> Its feature is frozen on July 10.

It is not a feature, just an optimization. So if it works well, it can
be included, no? Just curious. Maybe I am taking the 'feature freeze'
too literally?

--
Anurag Priyam

#7 - 10/31/2011 11:27 PM - funny_falcon (Yura Sokolov)

Made some changes to load.c patch https://gist.github.com/1272991 :

- made it configurable with RUBY_LOADED_FEATURES_SORTED environment variable (0 - disables sorting)
- redefine more $LOADED_FEATURES methods
- made patch against 1.9.3-p0

#8 - 11/01/2011 06:47 AM - funny_falcon (Yura Sokolov)

Patch applies on trunk cleanly.

#9 - 11/01/2011 06:54 AM - funny_falcon (Yura Sokolov)

I have a report from one man about it's application startup time with and without patch:

$ time ./bin/rails runner 'puts'

1.9.3-p0 without patch:
real 0m22.683s
user 0m21.417s
sys 0m0.956s

with patch:
real 0m17.806s
user 0m16.633s
sys 0m0.892s

#10 - 11/01/2011 08:23 AM - now (Nikolai Weibull)

On Mon, Oct 31, 2011 at 22:54, Yura Sokolov funny.falcon@gmail.com wrote:

> I have a report from one man about it's application startup time with and without patch:
>
> $ time ./bin/rails runner 'puts'

> 1.9.3-p0 without patch:
> real 0m22.683s
> user 0m21.417s
> sys 0m0.956s
>
> with patch:
> real 0m17.806s
> user 0m16.633s
> sys 0m0.892s

Really? Sounds way too good.

I can't help but wonder what the times would be if a hash was used
instead of a sorted array.

#11 - 11/01/2011 10:12 AM - trans (Thomas Sawyer)

"After this patch loaded_features is kept sorted by basename without extension (/usr/lib/ruby/asdf.rb => asdf). When rb_feature_p start its check, it goes straight to the first item with matching basename (using binary search) and stops after last."

If I understand this correctly, this is not good behavior. See http://redmine.ruby-lang.org/issues/4969.

If you guys are worried about load speed, there are two things that can be done that will speed up things more than anything else. 1) use relative require in your libraries. That's easy, but it's up to the end developer. And 2) Allow require/load methods to specify the specific gem in which a file is to be found and keep an index of the gem locations. This later approach can be done in one of two ways, either by a) taking the first part of require path to be the gem name by default, or b) allowing the gem name as an option, e.g. require 'foo', :from=>'foogem'. Ruby's standard library can use 'ruby' as a special "gem" name.

#12 - 11/01/2011 10:14 AM - trans (Thomas Sawyer)

Links broke for some reason, try again: http://redmine.ruby-lang.org/issues/4969

#13 - 11/01/2011 02:40 PM - funny_falcon (Yura Sokolov)

Thomas Sawyer wrote:

> "After this patch loaded_features is kept sorted by basename without extension (/usr/lib/ruby/asdf.rb => asdf). When rb_feature_p start its check, it goes straight to the first item with matching basename (using binary search) and stops after last."
>
> If I understand this correctly, this is not good behavior. See http://redmine.ruby-lang.org/issues/4969

Patch preserves require behaviour because it doesn't remove any checks against path or extension.
#4969 remains as it should, cause require system or ruby 1.9 were created to prevent such cases (if I understand correctly). One should put file in a unambiguous path (lib/wedge/abbrev.rb) and then require from there (require 'wedge/abbrev')

> If you guys are worried about load speed, there are two things that can be done that will speed up things more than anything else. 1) use relative require in your libraries. That's easy, but it's up to the end developer.

In fact, be present relative require is twice slower (in 1.9.2, 1.9.3), cause ruby checks such file before and after path expansion.

#14 - 11/01/2011 03:23 PM - now (Nikolai Weibull)

On Tue, Nov 1, 2011 at 06:40, Yura Sokolov funny.falcon@gmail.com wrote:

> In fact, be present relative require is twice slower (in 1.9.2, 1.9.3), cause ruby checks such file before and after path expansion.

Slower than what? Require_relative is a lot faster than require when
RubyGems is involved, as require_relative doesn't walk $LOAD_PATH when
it looks for the file.

#15 - 11/01/2011 03:35 PM - funny_falcon (Yura Sokolov)

> Slower than what? require_relative is a lot faster than require

Sorry, you are right, I confused require_relative 'somelib' with require './somelib'.
require_relative still gains improvement from this patch, cause it use require, but improvement is a bit lesser cause prepended path allows to skip some checks here http://redmine.ruby-lang.org/projects/ruby-trunk/repository/entry/load.c#L158

#16 - 11/01/2011 03:39 PM - funny_falcon (Yura Sokolov)

> require_relative still gains improvement from this patch, cause it use require, but improvement is a bit lesser cause prepended path allows to skip some checks here http://redmine.ruby-lang.org/projects/ruby-trunk/repository/entry/load.c#L158

Excuse me, I mean a line above http://redmine.ruby-lang.org/projects/ruby-trunk/repository/revisions/33027/entry/load.c#L157

#17 - 11/01/2011 04:23 PM - now (Nikolai Weibull)

On Tue, Nov 1, 2011 at 07:35, Yura Sokolov funny.falcon@gmail.com wrote:

> Issue #5427 has been updated by Yura Sokolov.
>
> > Slower than what? require_relative is a lot faster than require
>
> Sorry, you are right, I confused require_relative 'somelib' with require './somelib'.
> require_relative still gains improvement from this patch, cause it use require, but improvement is a bit lesser cause prepended path allows to skip some checks here http://redmine.ruby-lang.org/projects/ruby-trunk/repository/entry/load.c#L158

I don't think Thomas was talking about speed change in
require_relative thanks to your suggested changes, rather that one
should use require_relative regardless of your patch, as
require_relative is a lot faster.

#18 - 11/01/2011 10:24 PM - funny_falcon (Yura Sokolov)

Nikolai Weibull wrote:

> Yura Sokolov wrote:
>
> > Sorry, you are right, I confused require_relative 'somelib' with require './somelib'.
> > require_relative still gains improvement from this patch
> > I don't think Thomas was talking about speed change in require_relative thanks to your suggested changes,
> > rather that one should use require_relative regardless of your patch, as require_relative is a lot faster.

Sorry again, I thought it was clear when I said "sorry" first time I admit this fact.
Thank you for reprimand.

#19 - 11/01/2011 10:59 PM - now (Nikolai Weibull)

On Tue, Nov 1, 2011 at 14:24, Yura Sokolov funny.falcon@gmail.com wrote:

> Nikolai Weibull wrote:
>
> > I don't think Thomas was talking about speed change in require_relative thanks to your suggested changes
>
> Sorry again, I thought it was clear when I said "sorry" first time I admit this fact.

Ah, OK.

> Thank you for reprimand.

That sounds a bit harsh :-).

#20 - 12/08/2011 04:24 PM - funny_falcon (Yura Sokolov)

Currenly, I've update patch with fix from #5727
https://gist.github.com/1272991

#21 - 12/14/2011 05:59 PM - funny_falcon (Yura Sokolov)

- File load.c.patch added

I update patch against ruby-trunk.

Currently I could claim 6% loading time improvement on rails application with $LOADED_FEATURES.size near 1000.
But gain should exponentially increase with application size grow.

Patch attached. Pull request is here https://github.com/ruby/ruby/pull/66 .

#22 - 12/14/2011 06:53 PM - now (Nikolai Weibull)

On Wed, Dec 14, 2011 at 09:59, Yura Sokolov funny.falcon@gmail.com wrote:

> Currently I could claim 6% loading time improvement on rails application with $LOADED_FEATURES.size near 1000.

That's rather meager results (and actual timings say a lot more than
percentages).

What are the results for smaller $LOADED_FEATURES sizes (say, in the
0..99 range)?

> But gain should exponentially increase with application size grow.

Exponentially?

And how much larger of a $LOADED_FEATURES will we ever see?

#23 - 12/14/2011 11:20 PM - funny_falcon (Yura Sokolov)

> That's rather meager results (and actual timings say a lot more than percentages).

Yes, I know. Gain were much greater before #5727 were fixed. So that, caching expanded_load_path should be next step of optimization.

> What are the results for smaller $LOADED_FEATURES sizes (say, in the 0..99 range)?

Running 50 times ruby -r sequel -e 'exit' is 16.2sec without patch and 16.7sec with patch.
So that, it seems, that patch hurts performance by 0.01sec on each run (or, about 3%) when $LOADED_FEATURES is about 66.

> > But gain should exponentially increase with application size grow.
> > Exponentially?

Well, without patch complexity of loading is O(n**2), with patch is O(n*log(n)), so that, gain grows as n/log(n) :)
You eat me :)

> And how much larger of a $LOADED_FEATURES will we ever see?

It seems that 1600 is not uncommon, a man at #5703 has such with moderate application.

#24 - 12/14/2011 11:32 PM - funny_falcon (Yura Sokolov)

> Well, without patch complexity of loading is O(n**2), with patch is O(n*log(n)), so that, gain grows as n/log(n) :)

Ough, it should be O(n(n-log(n)) => O(n**2) %(

#25 - 12/16/2011 05:44 PM - funny_falcon (Yura Sokolov)

- File load.c.patch added

Tiny fix to patch: fix startup speed on small script.

#26 - 03/31/2012 10:29 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to nobu (Nobuyoshi Nakada)

Sorry I don't catch up the discussion.

What's the status?
The proposal is COMPLETELY compatible?
It may be helpful for me to create a short summary of the proposal and discussion.

I tentatively assign this to nobu.
Any other committer is interested in this issue?

--
Yusuke Endoh mame@tsg.ne.jp

#27 - 03/31/2012 07:58 PM - funny_falcon (Yura Sokolov)

> The proposal is COMPLETELY compatible?

Unlike #5767 , this proposal is slightly incompatible, because it forces some kind
of sort order on LOADED_FEATURES, and this has impact on debugging (one couldn't
know which file were required last). But it allows to disable itself by setting
environment variable 'RUBY_LOADED_FEATURES_SORTED=0'.

Also, some methods, which could change order of LOADED_FEATURES entries are undefined
at all (reverse!, sort!, shuffle!, rotate!, etc), but there is no one real application call
this methods.

So that, except debugging messages, this proposal fully compatible with any real
application.

Big applications gains about 6% of startup time improvement (whether #5767 applied or not).
Tiny script has neither benefits nor impact from this.

I believe, there could be another way to improve search across LOADED_FEATURES.
For example, instead of sort LOADED_FEATURES itself by basename of file, there
could be managed hash mapping basename to array of files. But this adds a lot of
complexity.

Yura aka funny_falcon

#28 - 11/20/2012 09:32 AM - h.shirosaki (Hiroshi Shirosaki)

- Status changed from Assigned to Closed

Fixed by #7158. See #7387.

Files

load.c.patch	6.8 KB	12/14/2011	funny_falcon (Yura Sokolov)
load.c.patch	6.67 KB	12/16/2011	funny_falcon (Yura Sokolov)