Ruby trunk - Feature #5767

Cache expanded_load_path to reduce startup time

12/15/2011 06:18 PM - funny_falcon (Yura Sokolov)

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

Description

This patch add caching of expanded load path.
It reduces rails startup time by 33%
(patch against 1.9.3-p0 and simple performance test is here https://gist.github.com/1480404)

History

#1 - 12/15/2011 07:23 PM - now (Nikolai Weibull)

On Thu, Dec 15, 2011 at 10:18, Yura Sokolov funny.falcon@gmail.com wrote:

> Issue #5767 has been reported by Yura Sokolov.
>
> ---
>
> Feature #5767: Cache expanded_load_path to reduce startup time
> http://redmine.ruby-lang.org/issues/5767
>
> Author: Yura Sokolov
> Status: Open
> Priority: Normal
> Assignee:
> Category:
> Target version:
>
> This patch add caching of expanded load path.
> It reduces rails startup time by 33%
> (patch against 1.9.3-p0 and simple performance test is here https://gist.github.com/1480404)

Is it only me, or is treating $LOAD_PATH like an array hurting
possible optimizations? I mean, how sane is it to allow
$LOAD_PATH.map!{ … }?

#2 - 12/15/2011 07:53 PM - shyouhei (Shyouhei Urabe)

Nikolai Weibull wrote:

> On Thu, Dec 15, 2011 at 10:18, Yura Sokolov funny.falcon@gmail.com wrote:
>
> > Issue #5767 has been reported by Yura Sokolov.
> >
> > ---
> >
> > Feature #5767: Cache expanded_load_path to reduce startup time
> > http://redmine.ruby-lang.org/issues/5767
> >
> > Author: Yura Sokolov
> > Status: Open
> > Priority: Normal
> > Assignee:
> > Category:
> > Target version:
> >
> > This patch add caching of expanded load path.
> > It reduces rails startup time by 33%
> > (patch against 1.9.3-p0 and simple performance test is here https://gist.github.com/1480404)
>
> Is it only me, or is treating $LOAD_PATH like an array hurting
> possible optimizations? I mean, how sane is it to allow
> $LOAD_PATH.map!{ … }?

$LOAD_PATH determines the order to search required files. At least it must be ordered linearly.

#3 - 12/15/2011 08:47 PM - funny_falcon (Yura Sokolov)

Nikolai Weibull wrote:

> Is it only me, or is treating $LOAD_PATH like an array hurting
> possible optimizations? I mean, how sane is it to allow
> $LOAD_PATH.map!{ … }?

I think, you right. $LOAD_PATH and $LOADED_FEATURES should be restricted.
But currently they not.
When I trying to improve $LOADED_FEATURES at #5427, I remove dangerous methods.

In this patch, it just invalidates cache when complex self modifying method is called,
and keeps cache valid in case of simple methods, like push, pop, <<

#4 - 12/16/2011 04:10 PM - funny_falcon (Yura Sokolov)

- File cache_expanded_load_path.patch added

Cosmetic update to patch: more atomic cache access.

#5 - 12/17/2011 09:07 PM - funny_falcon (Yura Sokolov)

- File find_file_safe_and_cache.patch added

#6 - 12/17/2011 09:12 PM - funny_falcon (Yura Sokolov)

There is report about 47% startup time reduction https://gist.github.com/1480404#gistcomment-69769

#7 - 12/18/2011 01:03 AM - luislavena (Luis Lavena)

For those interested here are some raw performance numbers and comparison between default (no patch) and other patches combined:

https://gist.github.com/1490555

#8 - 12/19/2011 04:07 AM - funny_falcon (Yura Sokolov)

- File cache_expanded_load_path.patch added

#9 - 12/19/2011 03:57 PM - funny_falcon (Yura Sokolov)

- File cache_expanded_load_path.patch added

#10 - 01/18/2012 07:14 PM - funny_falcon (Yura Sokolov)

I removed messing with file.c.
Instead rb_get_load_path returns expanded paths when optimization is enabled.

I have updated pull request, so that it contains one commit.

https://github.com/ruby/ruby/pull/68.patch
https://github.com/ruby/ruby/pull/68

#11 - 02/04/2012 02:54 PM - funny_falcon (Yura Sokolov)

https://gist.github.com/1688857

Most of success of that gist is cause of this issue.
Cites from comments:

Thanks for sharing that. Indeed the speed increase is quite impressive! I've tested the time it take to run some rspec test before and after. Here are the results:

Before: Finished in 402.79 seconds (3521 examples, 3 failures, 39 pending)
After: Finished in 222,82 seconds (3521 examples, 3 failures, 39 pending). WOOW!

Obligatory benchmarks:

Test Suite on 1.9.3-p0 proper : 2m06s
Rails boot time : 9.1s

Suite on 1.9.3-p0-perf : 1m32s
Rails boot time : 6.5s

Not bad.

$ time rake

before:
real 0m37.919s
user 0m6.474s
sys 0m0.919s

after:
real 0m34.236s
user 0m3.367s
sys 0m0.848s

$ time rails runner "Time.now"

before:
real 0m4.589s
user 0m3.871s
sys 0m0.629s

after:
real 0m2.437s
user 0m1.889s
sys 0m0.534s

5.5 secs instead of 8.8 secs for rails runner "puts \"omfg\""
rspec . 15.1 seconds instead of 18.2 seconds

One of my Rails apps, boot time (measured with ./bin/rails runner 'puts 3'

```
1.9.3-perf:  5.1 seconds
1.9.3-p0:    8.1 seconds
1.9.2-p290: 18.2 seconds
```

My rake parallel:spec went down from 68 to 38sec!!!!!!!!!! That's 44% cut off! Fantastic! THANK YOU!

API spec tests ~30% faster!

---

time bundle exec rake environment

Before:
14.60s user 1.26s system 100% cpu 15.861 total

After:
5.11s user 1.10s system 99% cpu 6.212 total

60.8% speedup!!!!!!

#12 - 03/31/2012 10:28 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

#13 - 03/31/2012 07:03 PM - funny_falcon (Yura Sokolov)

> The proposal is COMPLETELY compatible?

This proposal is completely compatible, since LOAD_PATH remains the same array, methods are overridden only to keep cache in sync.

Proposal is about keeping result of expanding LOAD_PATH entries in a 'invisible' cache.
This cache is lazy invalidated on changing directory and filesystem encoding, so that
the whole semantic of LOAD_PATH remains the same.
Methods of LOAD_PATH overridden to keep cache in sync, so that there is no need to expand
all paths again if entries were only appended to any side of LOAD_PATH, or removed from any side.

It saves up to 35% of big application's startup time and still brings slight improvement
for small scripts.

(Only Jeremy Evan's sequel use self crafted super fast load system which does not gain improvement from this patch)

I didn't check it against current trunk, but believe there could not be big problems.

There is still small room for improvement this patch: make chached expanded paths class to be
subclass of String, and then test against it in a file_expand_path, so that to not expand it
twice in calls from rb_find_file_ext_safe and rb_find_file_safe.

Yura aka funny_falcon

#14 - 03/31/2012 07:23 PM - trans (Thomas Sawyer)

Does this have any effect on https://bugs.ruby-lang.org/issues/4969 (scroll down to post #7).

#15 - 04/01/2012 12:04 AM - funny_falcon (Yura Sokolov)

> Does this have any effect on https://bugs.ruby-lang.org/issues/4969 (scroll down to post #7).

Thomas Sawyer, this proposal has no any effect on #4969 because it doesn't change require semantic
in any way. It only improve performance.

I leave my comment at #4969 and I still take that point: #4969 is about making game more tangled but easier to win instead of playing by the rules, imho.

#16 - 11/20/2012 09:34 AM - h.shirosaki (Hiroshi Shirosaki)

- Status changed from Assigned to Closed

Fixed by #7158.

Files

cache_expanded_load_path.patch	5.67 KB	12/15/2011	funny_falcon (Yura Sokolov)
cache_expanded_load_path.patch	6.88 KB	12/16/2011	funny_falcon (Yura Sokolov)
find_file_safe_and_cache.patch	5.34 KB	12/17/2011	funny_falcon (Yura Sokolov)
cache_expanded_load_path.patch	14.4 KB	12/19/2011	funny_falcon (Yura Sokolov)
cache_expanded_load_path.patch	15.4 KB	12/19/2011	funny_falcon (Yura Sokolov)