Ruby master - Feature #14492

iseq loading + caching should be in core

02/19/2018 04:14 AM - normalperson (Eric Wong)

Status:	Open	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	
Target version:	3.0	

Description

```
... And not in a RubyGem like yomikomu(*)

RubyGems itself is responsible for over 100ms of startup time on my system(**),
so it would be beneficial to make it part of core and speed up rubygems (and stdlib).
$ time ruby -e exit

real    0m0.160s
user    0m0.155s
sys     0m0.004s

$ time ruby --disable=gems -e exit

real    0m0.014s
user    0m0.013s
sys     0m0.000s

(*) git clone https://github.com/ko1/yomikomu.git

(**) I admit, I am intent on continuing use of Ruby on a laptop from 2005.
     That's roughly when I started using Ruby, so any hardware which Ruby
     worked well on back then should work equally well for current and future
     versions of Ruby.

     And maybe this summer I'll dig out a 600 MHz Duron from the early 2000s :D
```

Related issues:

Related to Ruby master - Feature #14489: MJIT needs a reusable cache	**Rejected**

History

#1 - 02/19/2018 09:40 AM - Eregon (Benoit Daloze)

FWIW, here are number on a laptop from 2014 with a i7-4702HQ @ 2.20GHz:

```
$ ruby -v
ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-linux]
$ time ruby -e exit
ruby -e exit  0.04s user 0.00s system 98% cpu 0.042 total
$ time ruby --disable-gems -e exit
ruby --disable-gems -e exit  0.00s user 0.00s system 94% cpu 0.005 total
```

So not such a big difference.

But I agree making RubyGems more lazy would be good.

#2 - 02/19/2018 12:07 PM - shevegen (Robert A. Heiler)

> So not such a big difference.

I also ran it just because I was curious.

> time ruby -e exit

real 0m0.109s

user 0m0.097s
sys 0m0.012s

time ruby --disable-gems -e exit

real 0m0.015s
user 0m0.005s
sys 0m0.010s

It is not a huge difference but almost noticable. Improvements there are probably useful simply because most ruby hackers may use gems and the gem ecosystem, so they may benefit. May not only be old hardware but also smaller systems where you may still use ruby.

It also somewhat fits to matz' theme/goal to make ruby faster, even if it may be a tiny benefit compared to e. g. gains achieved via mjit. Many tiny baby steps versus one big giant step - well, ideally both kind of steps together ... :)

#3 - 02/19/2018 09:12 PM - sam.saffron (Sam Saffron)

In general bootsnap has had significantly more dev time. The actual technique it uses does not require and c extensions (though it uses one for the the on-disk structure)

Agree we should have something like this built-in to Ruby, it would also allow us more flexibility around implementation.

One idea would be for gem authors to be allowed to also ship a .cache folder with ISeqs if they wish. (generating on demand has some security problems, since gem runner may not have permission to gem location)

Longer term I would like to see a way of jitting as well so the .cache folder also contains a jitted copy of the ISEQs. In theory we could ship platform specific .so files that encompass all the .rb files a gem has. This has tremendous saving around multiple file access and so on (auto loading and partial loading withstanding)

#4 - 02/21/2018 06:02 AM - ko1 (Koichi Sasada)

- Target version set to 3.0

I agree on it.

rubygems and did-you-mean are the first targets of this goal.

https://github.com/rubygsoc/rubygsoc/wiki/Ideas-for-MRI-(Matz-Ruby-Interpreter)#better-pre-compilation-support
is a step toward this goal.

BTW, one binary (like go-lang) is also extended goal.
For example, making one binary from Gemfile will be great.
Honestly it seems not so difficult. (than type system :p)

#5 - 02/24/2018 05:36 AM - k0kubun (Takashi Kokubun)

- Related to Feature #14489: MJIT needs a reusable cache added