Explore lazy RubyGems boot to reduce need for --disable-gems

02/02/2022 07:13 PM - headius (Charles Nutter)

Status: Open
Priority: Normal
Assignee: Headius
Target version: 

Description
In [https://bugs.ruby-lang.org/issues/17684](https://bugs.ruby-lang.org/issues/17684) there was debate about whether the --disable-gems flag should be removed. Several folks were in favor, since Ruby without RubyGems is fairly limited, but others wanted to keep the flag for small, fast command line scripts that do not depend on RubyGems.

Lazily loading RubyGems might be a middle ground, and it has been explored in some depth by TruffleRuby:


Eregon (Benoit Daloze) shows how this improves their startup time in this article from a couple years ago:

[https://eregon.me/blog/2019/04/24/how-truffleruby-startup-became-faster-than-mri.html](https://eregon.me/blog/2019/04/24/how-truffleruby-startup-became-faster-than-mri.html)

I believe this approach has merit and could be beneficial to both CRuby and JRuby if we can collaborate on how the lazy loading should happen and figuring out where the edges are. Eregon (Benoit Daloze) may know some of those edges if they have run into them in TruffleRuby.

A simple test of --disable-gems on CRuby 3.1 shows what an impact it has, which we might be able to duplicate in a lazy boot WITHOUT losing RubyGems functionality and default gem upgrading:

```
$ time ruby -e 1
real    0m0.107s
user    0m0.068s
sys     0m0.030s

$ time ruby --disable-gems -e 1
real    0m0.019s
user    0m0.007s
sys     0m0.008s
```

Over 80% of CRuby's base startup is due to eagerly booting RubyGems. We can do better!

Related issues:
Related to Ruby master - Feature #17684: Remove `--disable-gems` from release... Closed

History

#1 - 02/02/2022 07:19 PM - headius (Charles Nutter)
Another comparison: in JRuby, RubyGems booting accounts for around 1/3 of JRuby's startup time in our "fastest" mode. Given how poor our startup time is already, we are understandably interested in ways to speed it up.

#2 - 02/02/2022 07:20 PM - headius (Charles Nutter)
- Description updated

#3 - 02/02/2022 07:24 PM - headius (Charles Nutter)
Eregon (Benoit Daloze) I remember seeing code in TruffleRuby that built up an index of default gem files, so it would know which ones might need to trigger a gem load. I cannot find that now.

Could you provide some links to show how this is handled, and perhaps write up a quick description of how TruffleRuby avoids loading RubyGems but still supports upgrading default gems?

#4 - 02/02/2022 08:32 PM - Eregon (Benoit Daloze)

03/25/2022
#5 - 02/03/2022 12:48 PM - Eregon (Benoit Daloze)

I think a faster boot time while still having RubyGems enabled by default would be good for all Rubies.

https://github.com/oracle/truffleruby/blob/14ec2c2673188d47374a0570cf036864fcafe0b3/src/main/ruby/truffleruby/core/truffle/gem_util.rb is most of the logic for handling upgraded default gems.

And there is some logic directly in require and require_relative to handle upgraded default gems:


I wonder how much of this could actually move to RubyGems. Maybe RubyGems could be much more lazy and do something similar? That would probably make more sense as an issue on the RubyGems tracker though.

In Ruby 3.1.0, RubyGems loads 23 (rather big) Ruby files (4401 SLOC) + rbconfig and monitor (stringio, uri no longer, that was in 2.6).

In addition it also reads all default gem gemspecs (72 of them, 1422 SLOC), which might take a long time, and is more Ruby code loaded:

https://gist.github.com/eregon/5318509127d36e74dc5d559900760215

That's a lot, in comparison lazy rubygems loads 1 small Ruby file (158 SLOC), does not load any gemspecs, and it just checks if some paths exist for default gems in require.

To achieve that it's probably necessary to not define Gem early on and let that be an autoload (lots of state there that cannot easily be made lazy). So we would need another namespace module for the lazy-rubygems logic.

cc deivid (David Rodríguez)

#6 - 02/03/2022 02:21 PM - deivid (David Rodríguez)

I worked on https://github.com/rubygems/rubygems/pull/4199 a while ago which avoided all gemspec evaluation during boot if I recall correct, and made things faster. But I never wrapped it up, I'll try to make it happen and re-measure its impact.

#7 - 02/03/2022 04:17 PM - headius (Charles Nutter)

I think a faster boot time while still having RubyGems enabled by default would be good for all Rubies.

Thank you for providing those links... I knew I had seen that code but could not remember where it was located.

Making these changes in RubyGems would be great, if that is sufficient to reduce boot times. I was not clear on how much of this could live there versus how much needs to be changed in the Ruby boot process.

One thing that has always bothered me about RubyGems is its constant rescanning of gemspecs and lib directories from gems which I think we all agree should NEVER BE MODIFIED. There's no good reason not to build up a local file index, a la gel. At worst, a gem reindex command that rescans all installed gemspecs and libs would be sufficient to support the very few dev-time cases for adding or removing files from an installed gem.

I worked on https://github.com/rubygems/rubygems/pull/4199 a while ago

This approach looks similar in spirit to gel in that it creates an index of each gem's contents. A single index might be even faster but would require different strategies to maintain it. The advantage of a single index would of course be a single file read, and potentially that file produces exactly the structure we need to quickly find required files.

I'm excited to see this moving forward. On @tenderlove's recommendation I am adding a note to the next dev meeting.

#8 - 02/03/2022 06:46 PM - Eregon (Benoit Daloze)

https://github.com/rubygems/rubygems/pull/4199

Interesting, that seems a helpful step, although I suspect there would still be a significant overhead over --disable-gems, as long as we load so much Ruby code during startup.

I think upgraded_default_gem? in https://github.com/oracle/truffleruby/blob/14ec2c2673188d47374a0570cf036864fcafe0b3/src/main/ruby/truffleruby/core/truffle/gem_util.rb#L57 has two advantages compared to that approach:

- It hardcodes the list of default gem names, which AFAIK is anyway not possible to change for a given Ruby release, but iterating the default directory for default gem names would likely be fast enough.
- It assumes default gems have sensible require-able files, so require 'foo/bar/...' if it's a default gem file must be for default gem foo, which holds for all current default gems AFAIK. Just the special case of gem names with a - in it like net/http, then we it checks if the path starts with net...
That avoids e.g. having a list of all files which is large memory footprint and rather time consuming to get.
The purpose of upgraded_default_gem? is to avoid loading RubyGems when loading non-gem stdlib files.

BTW I think such an optimization would be very useful for non-default gems too. Right now require 'not-a-default-gem' or require 'not-a-default-gem/foo.rb' loads all gemspecs (so it's O(installed gems), which is problematic with many gems installed), while it very likely is in a gem named not-a-default-gem. This can affect semantics for gems which don't respect the convention and might be earlier alphabetically than a gem with the proper name, not sure that matters though.

#9 - 02/04/2022 07:42 AM - byroot (Jean Boussier)
Another thing that could be explored would be to leverage RubyVM::InstructionSequence.compile_file(path). If we indeed have a mapping of the gems content, we could define load_iseq, at least for the duration of rubygems's "boot".

#10 - 02/04/2022 11:33 AM - Eregon (Benoit Daloze)
byroot (Jean Boussier) wrote in #note-9:
Another thing that could be explored would be to leverage RubyVM::InstructionSequence.compile_file(path). If we indeed have a mapping of the gems content, we could define load_iseq, at least for the duration of rubygems's "boot".

That would only work on CRuby, so as such would be unusable on anything but CRuby.
I think we should rather load less Ruby code while starting RubyGems.

#11 - 02/04/2022 11:49 AM - matthewd (Matthew Draper)
Over 80% of CRuby's base startup is due to eagerly booting RubyGems. We can do better!

It's not the main point here, but perhaps still worth noting: --disable-gems implies --disable-error_highlight,did you mean.

In my own brief local test, that makes the ratio more like 17% CRuby, 70% rubygems, 13% those two. *

* I do use Gel day-to-day, so I have approximately zero non-default gems installed. It's only two requires, but if you have a lot of gems, that would skew the ratio.

Gel does indeed maintain a single index file (well, one per gem installation target, so platform or ruby version for ext gems, and one for all plain-ruby gems). It lost a good portion of its speed when SDBM was kicked out of stdlib, and I had to fall back to PStore, though. (I'm currently experimenting with a load-everything-at-boot strategy, which makes it faster at the cost of more filename strings in memory. Without inventing a new native storage library, I imagine I'll ultimately need to handle custom storage through direct IO, but for now the extra memory is a tolerable trade-off.)

Constructing the index is not overly challenging: during gem installation, you're already responsible for copying the files into their new home, so adding them to an extra list is trivial. (Admittedly, that's less true for default gems.)

#12 - 02/04/2022 12:10 PM - byroot (Jean Boussier)

That would only work on CRuby, so as such would be unusable on anything but CRuby.
I think we should rather load less Ruby code while starting RubyGems.

It's not a "one or the other" scenario. We can perfectly explore both.

#13 - 02/04/2022 04:10 PM - Eregon (Benoit Daloze)
byroot (Jean Boussier) wrote in #note-12:

It's not a "one or the other" scenario. We can perfectly explore both.

Yes, but if we manage to slim RubyGems boot like lazy-rubygems does, it loads so little Ruby code that I think there is no point to try to load that faster as iseq.

#14 - 02/17/2022 07:15 AM - matz (Yukihiro Matsumoto)
Lazy loading RubyGems sounds great. Ping us if you need support from the core.

Matz.