Ruby master - Feature #17684
Remove `--disable-gems` from release version of Ruby

03/10/2021 12:51 PM - hsbt (Hiroshi SHIBATA)

<table>
<thead>
<tr>
<th>Status</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee</td>
<td></td>
</tr>
<tr>
<td>Target version</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

In my understand, `--disable-gems` is only debugging feature for ruby-core team.

But some users enabled its option in test environment for performance or etc. So, `--disable-gems` option is wrong usage for some users.

- [https://github.com/rubygems/bundler/issues/7487#issuecomment-569901549](https://github.com/rubygems/bundler/issues/7487#issuecomment-569901549)
- [https://github.com/rubygems/rubygems/pull/4440#issue-587031184](https://github.com/rubygems/rubygems/pull/4440#issue-587031184)

We should remove it from package version of ruby.

**Related issues:**
- Related to Ruby master - Feature #18376: Version comparison API [Open]
- Related to Ruby master - Feature #18568: Explore lazy RubyGems boot to reduce... [Open]

**Associated revisions**

Revision d58daad3 - 03/17/2021 06:36 AM - naruse (Yui NARUSE)

[Feature #17684] Declare `--disable-gems` is for debugging

Revision 2c0de392 - 12/13/2021 05:19 AM - mame (Yusuke Endoh)

NEWS.md: `--disable-gems` is now just for debugging. [Feature #17684]

**History**

#1 - 03/10/2021 12:59 PM - chrisseaton (Chris Seaton)

How should people who were using it for performance migrate off the option?

#2 - 03/10/2021 01:00 PM - naruse (Yui NARUSE)

Agreed. Unfortunately this increases the cost of development to support `--disable-gems`. In other words, this feature breaks the assumption that every Ruby has RubyGems. This feature should be removed.

#3 - 03/10/2021 01:02 PM - naruse (Yui NARUSE)

chrisseaton (Chris Seaton) wrote in [note-1]:

> How should people who were using it for performance migrate off the option?

`--disable-gems` should be just ignored in the future version, and the performance will be degraded.

#4 - 03/10/2021 01:21 PM - chrisseaton (Chris Seaton)

I don't have anything invested in this myself, but just for information to help the discussion, here's two examples of apps that use the option

- [https://github.com/Shopify/shopify-app-cli/blob/master/bin/shopify](https://github.com/Shopify/shopify-app-cli/blob/master/bin/shopify)

#5 - 03/10/2021 01:35 PM - xtkoba (Tee KOBAYASHI)

Will we be able to re-enable the `--disable-gems` option when we build ruby from source code? I think it is useful for narrowing problems down. For that purpose it might be sufficient that miniruby does not depend on Gems, though.

For performance we can use mruby ([https://github.com/mruby/mruby](https://github.com/mruby/mruby)) after all.
I'd love to see this clarified. IMO, there should be other ways to manage $LOAD_PATH then RubyGems. For such use cases, this option is useful IMO.

I disagree on removing this option, I don't think the benefits of removing it are strong enough, even if it didn't make any difference in performance.

I'm also using the --disable-gems option. I make games on Ruby but I'm not using gems and since the main audience is Windows, I have to distribute portable Ruby version that HAS to work on any path (including those with accents). I noticed that the rubygems are causing a lot of issues when the game needs to load native extensions (FMOD, SFML, sockets...) so I disable the gems and if they're needed, we include them explicitly using the require 'rubygems' command.

I would suggest that people who needs gems require rubygems manually or that ruby load rubygems automatically only if the script has a Gemfile (I bet this might already be handled by bundle & the bundle exec command).

I think --disable-gems is also useful to investigate issues, e.g. RubyGems used to override Kernel#warn (fixed in Ruby 3). It can significantly simplify investigating issues with require.

I agree it's not very common to need it for debugging, but it turned up useful a couple times. If it's not available for releases then it becomes inconvenient to investigate such cases.

Also the impact on startup time is still significant, so for very short scripts it can be noticeable (especially if calling out to subprocesses or running many small scripts).

That's something that can potentially be improved in CRuby and/or RubyGems. For instance by loading RubyGems lazily (autoload :Gem, try loading rubygems the first time require fails), which TruffleRuby does.

Finally, I guess for JITs it's quite useful to limit the amount of Ruby code loaded and run on startup, so having --disable-gems to investigate JIT issues seems useful.

If it only works on development builds, then again it's inconvenient to investigate a bug report from a user using a release.

I like using --disable-gems for scripts that need to be fast. For example, I call ruby from Vim to get some information about the Ruby environment:

```
" Add stdlib of environment's ruby to path
let g:stdlib = system('ruby --disable-gems -rrbconfig -e"print RbConfig::CONFIG["rubylibdir"]"')
let &path .= "," . stdlib
let g:ruby_path = &path
```

Adding 60ms (or more) for these "fast commands" would be pretty annoying.

```
[aaron@tc-lan-adapter ~]$ time ruby --disable-gems -rrbconfig -e"print RbConfig::CONFIG["rubylibdir"]"
/Users/aaron/.rubies/ruby-trunk/lib/ruby/3.1.0
```

```
+ Executed in 20.60 millis  fish  external
+  usr time  12.12 millis  146.00 micros  11.98 millis
+  sys time   6.33 millis   741.00 micros   5.59 millis
```

```
[aaron@tc-lan-adapter ~]$ time ruby -rrbconfig -e"print RbConfig::CONFIG["rubylibdir"]"
/Users/aaron/.rubies/ruby-trunk/lib/ruby/3.1.0
```

```
+ Executed in  79.02 millis  fish  external
+  usr time  61.54 millis  115.00 micros  61.43 millis
+  sys time  14.46 millis   679.00 micros  13.78 millis
```

I also have a use case where I use --disable-gems for performance. I maintain a tool that provides some shell hooks in ruby, and any slow down in getting to a prompt is incredibly noticeable.

Another use case: Bundler's --standalone mode knows all gem directories and doesn't need to load either RubyGems or Bundler, which can save quite some time on application startup. It seems a shame if it would still load RubyGems needlessly due to removing --disable-gems.
I forgot another place I use --disable-gems. I think RubyGems does lots of IO at startup, and IO is extremely slow on my Raspberry Pi. I’m using Ruby on Raspberry Pi for some IoT stuff around my house and I use --disable-gems in those environments. Here is the difference:

```bash
pi@lcd-screen:~ $ time ruby -v --disable-gems -e '
    ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [armv6l-linux-eabihf]

real  0m0.291s
user  0m0.118s
sys   0m0.082s

pi@lcd-screen:~ $ time ruby -v -e '
    ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [armv6l-linux-eabihf]

real  0m2.395s
user  0m1.998s
sys   0m0.129s
```

My Ruby installation only has the default gems installed, but RubyGems adds 2 seconds to startup.

---

My gut feeling is that it's miserable for people to confuse --disable-gems as a zero-cost speed boost option. Maybe rubygems is too slow now.

---

Status changed from Open to Closed

---

[Feature #17684] Declare --disable-gems is for debugging

At this time we just update --help.

The conclusion of ruby core's discussion is:

- As the first step, let's update the --help to explicitly state that --disable-gems is just for debugging
- If rubygems startup becomes so fast in future, we can consider removal of --disable-gems (but still it is desirable for debugging)
- Or someone who happen to report about --disable-gems again, we will consider again.

We also discuss RubyGem accesses too many files in startup. We expect gem team improve that.

https://gist.github.com/ko1/f0d8c899d580f7f69bb5608ab6d8215f

---

Related to Feature #18376: Version comparison API added

---

Related to Feature #18568: Explore lazy RubyGems boot to reduce need for --disable-gems added