Ruby - Bug #21022

The --with-modular-gc= option is cumbersome

01/09/2025 04:34 PM - vo.x (Vit Ondruch)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.4.1 (2024-12-25 revision 48d4efcb85) +PRISM +GC [x86_64-linux]	**Backport:**	3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN, 3.4: UNKNOWN

Description

Just experimenting a bit with the --with-modular-gc= if it is worth of enabling in Fedora, but I somehow cannot wrap my head around that. What is the intended configuration?

From Fedora POV, we have Ruby installed somewhere in /usr, which can only be managed by RPM. I don't think that we would like to package any additional GC. But if we wanted, we would likely installed it outside of the default Ruby directory structure. We would also like our users to experiment. So they would ideally have their GC built somewhere in their HOME. In that case, I'd expect that something like RUBY_GC_LIBRARY=/home/johndoe/path/to/gc/gc.so ruby -e 'puts "Hello custom GC"' would be command to run. But it does not seems that the current configuration option / env variables have anything like this in their mind.

Actually I wonder why the GC is not packaged as gem? Maybe RubyGems are late to the party to load the GC early enough. But they still would be nice method of distribution.

From #20860 I understand that the MMTk is experimental. But from the Ruby 3.4, the Modular GC seems to be as something users might want.

History

#1 - 01/09/2025 04:34 PM - vo.x (Vit Ondruch)

- Subject changed from The --with-modular-gc= option is a to The --with-modular-gc= option is cumbersome

#2 - 01/09/2025 05:50 PM - peterzhu2118 (Peter Zhu)

> What is the intended configuration?

We realize that this feature is currently cumbersome to use as it requires a compile time option, an environment variable, and a built GC library. We need the compile time option as there is a performance penalty when modular GC is enabled, as shown in the benchmarks in #20470. This is because the GC is no longer statically compiled and we use function pointers instead, which prevents compiler optimizations.

> I'd expect that something like RUBY_GC_LIBRARY=/home/johndoe/path/to/gc/gc.so ruby -e 'puts "Hello custom GC"' would be command to run.

That was the original design, however, in [Feature #20470] it was decided that this was a security concern due to the possibility to execute arbitrary code using environment variables. The compromise is to specify a directory at build time using --with-modular-gc=, and only accept binaries placed in that directory.

> Actually I wonder why the GC is not packaged as gem? Maybe RubyGems are late to the party to load the GC early enough. But they still would be nice method of distribution.

That is the the ultimate goal. We will probably need to do modifications to bundler to get this to work.

> But from the Ruby 3.4, the Modular GC seems to be as something users might want.

The feature is experimental, so we don't enable it for the vast majority of the users who don't want it.

#3 - 01/10/2025 09:59 AM - vo.x (Vit Ondruch)

peterzhu2118 (Peter Zhu) wrote in #note-2:

> What is the intended configuration?

> We realize that this feature is currently cumbersome to use as it requires a compile time option, an environment variable, and a built GC library. We need the compile time option as there is a performance penalty when modular GC is enabled, as shown in the benchmarks in #20470.

That mentions 1-2 %. Assuming that the ultimate target is to be able to use different GC, which in theory might compensate for the performance hit, I think that could be sensible trade off

BTW in that ticket, there is also mentioned "loading Ruby's current GC using RUBY_GC_LIBRARY_PATH". Should I have the default GC available somewhere as some .so file when compiling with the --with-modular-gc option?

> This is because the GC is no longer statically compiled and we use function pointers instead, which prevents compiler optimizations.

> > I'd expect that something like RUBY_GC_LIBRARY=/home/johndoe/path/to/gc/gc.so ruby -e 'puts "Hello custom GC"' would be command to run.

> That was the original design, however, in [Feature #20470] it was decided that this was a security concern due to the possibility to execute arbitrary code using environment variables. The compromise is to specify a directory at build time using --with-modular-gc=, and only accept binaries placed in that directory.

Fedora is binary distribution. Therefore having build time option is not helpful for our users. Especially if it enables to specify just one path. E.g. if we said that the Ruby GCs on Fedora will be placed in lets say /usr/lib64/rubygc, then that means this place should be managed by RPM and it would be less then ideal if users placed there their content. If it was placed in /usr/local/lib64/rubygc, then on contrary, RPM should not manage content of that directory. But in general, users (including developers) should not touch these locations at all, because these are system wide and requires elevated privileges, which is mostly wrong. For development purposes, they should stay in their $HOME.

But anyway, given that is seems you are aiming to provide the alternative GCs as gem, then this is likely just academic discussion :)

> > Actually I wonder why the GC is not packaged as gem? Maybe RubyGems are late to the party to load the GC early enough. But they still would be nice method of distribution.

> That is the the ultimate goal. We will probably need to do modifications to bundler to get this to work.

> > But from the Ruby 3.4, the Modular GC seems to be as something users might want.

> The feature is experimental, so we don't enable it for the vast majority of the users who don't want it.

Ok. Thanks for elaborating. It seems that ATM, it will be better to keep this option disabled for Fedora.

Thanks a lot for all the details and feel free to close this ticket.

#4 - 01/10/2025 03:23 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

#5 - 01/10/2025 03:27 PM - peterzhu2118 (Peter Zhu)

- Status changed from Closed to Open

> That mentions 1-2 %. Assuming that the ultimate target is to be able to use different GC, which in theory might compensate for the performance hit, I think that could be sensible trade off

Right, I agree that it's a small performance penalty. However, we don't have another production ready GC implementation yet, so for most users, they won't be able to take advantage of this feature and will instead just be taking a 1-2% performance penalty for no reason.

> BTW in that ticket, there is also mentioned "loading Ruby's current GC using RUBY_GC_LIBRARY_PATH". Should I have the default GC available somewhere as some .so file when compiling with the --with-modular-gc option?

There will always be a copy of the default GC baked into the Ruby binary, so even if you compiled with --with-modular-gc and didn't specify a RUBY_GC_LIBRARY environment variable, it will load the default GC.

> Fedora is binary distribution. Therefore having build time option is not helpful for our users.

We're aware that this is not the ideal design, and this is on purpose. As I've stated above, this is an experimental feature and is not production ready, so we make it optional until we have a faster and more stable implementation. We'll work on a better integration and delivery method (e.g. through a

gem) once we have a performant and stable alternate GC implementation.

> It seems that ATM, it will be better to keep this option disabled for Fedora.

Absolutely. It's experimental and we turn it off by default because of the performance penalty and potential instability.

#6 - 01/10/2025 03:27 PM - peterzhu2118 (Peter Zhu)

- Status changed from Open to Closed