Ruby master - Bug #11490

Allocation tracer sometimes attributes allocations to the wrong source file/line

08/27/2015 07:38 AM - chancancode (Godfrey Chan)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:		**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

See the reproduction script in https://gist.github.com/chancancode/dc175e702c02cdfa5ffb

This was originally brought to my attention via https://github.com/skylightio/skylight-ruby/issues/36

As you can see in the Github ticket, we overwrote Kernel#require and was incorrectly blamed for allocating a lot of objects/memory. I checked the source and saw that the file/line info is filled by the last Ruby-land control frame, so I was almost ready to accept this as an unfortunate artifact that cannot be fixed.

However, when I made the standalone reproduction script to test my theory, I noticed that it actually gets it right sometimes (the lines that reports nokogiri.bundle:0 instead of allocation_tracker_bug.rb:11), so it seems like something else is going on here.

By the way, since RubyGems overwrites Kernel#require, even without any other third-party libraries loaded, you can already observe allocation tracer attributing a lot of allocations to RubyGem's kernel_require.rb out of the box.

History

#1 - 08/27/2015 07:42 AM - chancancode (Godfrey Chan)

Related: when the allocation tracer is trying to blame the Kernel#require monkey patch, the line number appears to be off-by-one ("11" instead of "12")

#2 - 08/27/2015 09:10 PM - chancancode (Godfrey Chan)

Thinking about it more, it seems my original hunch is *probably* correct: when loading/parsing/evaluating the file, the VM has to allocate a large amount of objects (e.g. when it runs def zomg; ...; end, it needs to create the "immortal"symbol :zomg, among other things like InstructionSequence, etc). It then attributed the allocation of these objects to the closest Ruby frame. Since the "raw" Kernel#require is implemented in C, whoever calls it will end up "getting the blame".

That is fine in an ideal world – this would point to the line where you required the file with lots of code (e.g. the require "nokogiri" line), and that is at least somewhat useful (I can save X MB if I don't use nokogiri, and you can choose to make that tradeoff; although that is rarely a real problem). Unfortunately, **all** Rubies (C Ruby) have at least one Kernel#require override in Ruby-land (in RubyGems), so in practice, this information is always going to point you to the "wrong" place. Most commonly, it will be this line in RubyGems, but it could also be this line in Active Support or a similar place in bundler.

I'm not sure what's the best way to go about fixing this. Perhaps it should not report any allocation source/line for these internal VM things (the symbols, instruction sequence, etc), report it as "(VM internal):0", or perhaps point it to the corresponding line in the source (:zomg is "allocated" by the def zomg line).

Or perhaps it is just not a big deal and not worth the effort. If my explanation is right, I can accept that this is technically "correct", but evidently people have tried to use this information to "slim down" their app and this has caused them to go down the wrong rabbit hole.

#3 - 09/01/2015 09:02 AM - ko1 (Koichi Sasada)

- Status changed from Open to Feedback

- Assignee set to ko1 (Koichi Sasada)

I don't think "(VM internal):0" is useful.

#4 - 08/13/2019 04:18 AM - jeremyevans0 (Jeremy Evans)

- Status changed from Feedback to Closed

Files

allocation_tracker_bug.rb	786 Bytes	08/27/2015	chancancode (Godfrey Chan)