Ruby trunk - Bug #15625

Module#name performance has exponential-time worst case by aliased constants

02/27/2019 05:36 PM - nelhage (Nelson Elhage)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.6.1p33 (2019-01-30 revision 66950) [x86_64-darwin18]	**Backport:**	2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN

Description

It's well-known (c.f #11119) that Module#name has poor performance on anonymous classes, due to searching the entire constant namespace of the VM.

However, we recently discovered that it's even worse than that. In the presence of aliased constants (e.g. module M; end; Alias = M), the find_class_path method actually searches every *path* to a given module (e.g. it will visit that module under both the M and Alias names). This can lead to exponential-time behavior in cases of repeated aliases. See the attached test case, where performance gets twice as slow with every additional layer of aliasing:

On my laptop:

```
       user     system      total        real
before  2.183899   0.012151   2.196050 (  2.355187)
       user     system      total        real
a19  5.614596   0.024394   5.638990 (  5.702878)
       user     system      total        real
a10  9.204399   0.052817   9.257216 (  9.391010)
       user     system      total        real
a11 16.184035   0.060854  16.244889 ( 16.561101)
```

Our house style makes extensive use of aliases as an import-like mechanism to provide short names within a given scope. We discovered this bug while exploring an actual performance problem -- this is not just a theoretical report.

History

#1 - 02/27/2019 08:11 PM - shevegen (Robert A. Heiler)

This may be useful to mention at an upcoming developer meeting.

Aliased constants are probably quite common; I use them in a few of my projects myself, although this is very minor compared to me using aliased methods (I use a LOT of aliased methods; mostly because that way I can get more flexibility when tapping into "actionable calls" for different classes).

Perhaps it may be that there is not yet any good idea for what to change exactly, to improve this behaviour?

I had a look at the old report and suggestions by headius at https://bugs.ruby-lang.org/issues/11119 but I think the proposal may be too restrictive e. g. when it is suggested to disable the functionality. It reminds me a bit of the proposal to remove object_id, due to some problems associated with it, which I think was not an ideal trade-off (since it would also mean that people would lose functionality/flexibility, and this may be a trade-off / constraint).

Perhaps there could be some alternatives, if it is not trivial to improve it (speed-wise), to have the default as it is, but some additional way to use functionality without the speed trade off. This may not be directly related to the functionality or bug talked about here, but we have seen the addition of refinements, and matz has mentioned at several points to wish to introduce some form of namespace system. Perhaps something similar could be thought about where people could, if they want to, and so desire to, pick alternatives that may not be as flexible as the current status quo, but would come with a speed improve, on a per "namespace" basis (or per module/class basis, to use current terminology).

Or perhaps you may have a completely different suggestion - I am aware that your suggestion is not the same as headius' one as such. Just mentioning a few things. :)

(It could be worth to suggest it for the upcoming developer meeting, if only to get one or the other comment about it from the ruby core/main team and of course matz, if there is enough time for this at the meeting.)

#2 - 03/04/2019 11:52 AM - nobu (Nobuyoshi Nakada)

- Subject changed from Module#name performance has exponential-time worst case to Module#name performance has exponential-time worst case by aliased constants

- File bug-15625.patch added

Of course it is possible to check duplicated paths to refine worst cases, but it is a trade-off for usual case; no aliased constants.

trunk

	user	system	total	real
before	2.117049	0.003000	2.120049	(2.123071)
a9	5.685656	0.006082	5.691738	(5.697850)
a10	9.309634	0.013036	9.322670	(9.336729)
a11	16.784519	0.049488	16.834007	(16.883028)

patched

	user	system	total	real
before	3.161939	0.055769	3.217708	(3.223522)
a9	3.317957	0.046668	3.364625	(3.369195)
a10	3.316404	0.045762	3.362166	(3.366455)
a11	3.321730	0.048193	3.369923	(3.374927)

#3 - 03/04/2019 01:13 PM - Eregon (Benoit Daloze)

Why is such a search performed every time?
Is it not enough to cache the name in the Module instance (since it keeps the name even after the constant is removed anyway) ?

Also, I think naming can be done proactively when assigning constants, instead of a lazy search (that's what TruffleRuby does).

Files

classname.rb	854 Bytes	02/27/2019	nelhage (Nelson Elhage)
bug-15625.patch	1.38 KB	03/04/2019	nobu (Nobuyoshi Nakada)