Ruby trunk - Bug #11119

Anonymous classes and modules have terrible #name and #inspect performance

05/04/2015 04:07 PM - headius (Charles Nutter)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	all versions	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

MRI lazily determines the name of a class or module by walking all defined constants starting from Object and looking for the namespace in question. This allows deferring the full name calclation until the class/module is finished being defined. However, if the class or module is *never* accessible via Object, then this system-walking occurs for every call to #name or #inspect on the class/module and every call to the default #inspect on instances of the class.

A simple benchmark:

```
require 'benchmark'

module B
  module X
  end
end

def a
  c = Class.new
  c2 = Class.new
  c.class_eval 'A = c2'
  c2.class_eval 'A = c'

  c
end

c = a
x = B::X

loop do
  puts 'named'
  puts Benchmark.measure { 1_000_000.times { x.name } }
  puts 'anon'
  puts Benchmark.measure { 1_000_000.times { c.name } }
  cobj = c.new
  puts 'anon obj'
  puts Benchmark.measure { 1_000_000.times { cobj.inspect } }
end
```

Results on MRI 2.2 and JRuby 1.7 HEAD:

MRI:

```
named
  0.210000   0.000000   0.210000 (  0.205585)
anon
 14.170000   0.050000  14.220000 ( 14.259003)
anon obj
 15.750000   0.060000  15.810000 ( 15.864806)
```

JRuby:

```
named
  0.250000   0.000000   0.250000 (  0.253000)
```

```
anon
  0.270000   0.000000   0.270000 (  0.264000)
anon obj
  0.450000   0.000000   0.450000 (  0.447000)
```

The effect worsens linearly with the size of the system. Running in a freshly-generated Rails app's console:

```
named
  0.260000   0.020000   0.280000 (  0.272182)
anon
240.900000   0.800000 241.700000 (242.384455)
anon obj
257.070000   1.110000 258.180000 (261.986562)
```

I believe MRI needs to give up on looking for the object after the first failed namespace traversal, or else eagerly build this name the way other implementations do (and accept some changes).

History

#1 - 05/04/2015 04:08 PM - headius (Charles Nutter)

- Subject changed from Anonymous classes and modules have exponentially worse #name and #inspect performance to Anonymous classes and modules have terrible #name and #inspect performance

#2 - 05/04/2015 04:29 PM - nobu (Nobuyoshi Nakada)

- Description updated

What kind of changes?

#3 - 05/04/2015 04:36 PM - headius (Charles Nutter)

Nobuyoshi Nakada wrote:

> What kind of changes?

The kind of changes that make it impossible to pass all current tests in JRuby :-)

- For namespaces with multiple parents, traversal order will change #name
- For unrooted anonymous namespaces, an eager #name may produce a class path that's not actually addressable (not rooted in Object)

All namespaces would get predictable names, but in some cases those names would be different. The naming basically has to become eager (set in constant declaration like JRuby and Rubinius), which means later hierarchical changes in the system won't be reflected in the name.

#4 - 05/04/2015 04:39 PM - headius (Charles Nutter)

A simple rule for future Rubies might be "if you want an anonymous module/class to have a useful path-based #name, ensure the parent namespace is itself named, all parents provide the same guarantee, and the namespace hierarchy is eventually rooted in Object."

We can also avoid the unrooted case by following this rule: if at first assignment, a given namespace is not being set into an Object-rooted namespace hierarchy, it remains unnamed and unaddressable forever.

#5 - 08/29/2015 01:04 PM - Eregon (Benoit Daloze)

The way I do it in JRuby+Truffle is by lazily associating names when constants are assigned.
Then "name" the value if it is a Module and the lexical parent has a name/is rooted
and in that case also iterate on that module constants and name the lexical children.

This seems to have quite good performance for all cases (traversal is done once at constant assignment time only for modules)
and be fairly compliant. Of course it might not have the exact same ordering but I did not meet any difference so far.

#6 - 01/17/2018 06:16 AM - brodock (Gabriel Mazetto)

Some new benchmarks with recent MRI versions:

Ruby 2.3.6:

```
named
  0.300000   0.010000   0.310000 (  0.305285)
anon
 31.550000   0.120000  31.670000 ( 32.200161)
anon obj
```

```
 33.820000   0.180000  34.000000 ( 34.864390)
```

Ruby 2.4.1:

```
named
  0.110000   0.000000   0.110000 (  0.110127)
anon
 21.670000   0.110000  21.780000 ( 22.361176)
anon obj
 23.370000   0.180000  23.550000 ( 24.838731)
```

Ruby 2.5.0:

```
named
  0.358641   0.001790   0.360431 (  0.371241)
anon
 67.509984   0.269998  67.779982 ( 69.150504)
anon obj
 69.621064   0.212278  69.833342 ( 70.606355)
```

This suggests it got better in 2.4.x branch but worst in 2.5.x

#7 - 05/22/2019 09:29 AM - byroot (Jean Boussier)

This was fixed in https://bugs.ruby-lang.org/issues/15765