Ruby - Feature #6668

Multiple assignment should not return an Array object

06/30/2012 02:11 AM - headius (Charles Nutter)

Status:	Rejected	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

Currently, when doing multiple assignment, the entire expression must return the right-hand side as an array.

system ~ $ ruby -e "ret = (a, b, c = 1, 2, 3); p ret"
[1, 2, 3]

This is an artifact of MRI's implementation, since multiple assignment was traditionally implemented by taking the array node on the right-hand side, standing it up as a full Ruby Array, and then peeling elements off for assignment on the left-hand side. It is also a performance issue, since it requires constructing the RHS array even when it is never used (unless you are able to do various compiler tricks). I propose removing it.

Justification:

- The feature is rarely used; most people don't even know it exists.
- The impact of creating the RHS array is significant; JRuby can optimize it away in cases where the line is not used as an expression, and the performance difference is huge: https://gist.github.com/3019255
- It is counter-intuitive to have an automatic performance hit just from grouping assignments. "a,b = 1,2" should have the exact same performance as "a = 1; b = 2"

Note that while JRuby can eliminate the array creation in non-expression cases, those are somewhat rare since many times masgn is used at the end of a method body, as for initializers:

class Foo
def initialize(a, b, c)
@a, @b, @c = a, b, c
end
end

JRuby and other implementations may get smart enough in our optimizers to eliminate the array in all cases where it's not needed, but this is a very large burden on the optimization subsystem. It may also not be possible to do in all cases (or not possible to do in even a majority of cases).

Multiple assignment should not return RHS as an array. I do not care what it returns.

History

#1 - 09/19/2012 04:08 PM - headius (Charles Nutter)

Ping!

#2 - 09/19/2012 07:53 PM - ko1 (Koichi Sasada)

(2012/09/19 0:08), headius (Charles Nutter) wrote:

> This is an artifact of MRI's implementation, since multiple assignment was traditionally implemented by taking the array node on the right-hand side, standing it up as a full Ruby Array, and then peeling elements off for assignment on the left-hand side. It is also a performance issue, since it requires constructing the RHS array even when it is never used (unless you are able to do various compiler tricks). I propose removing it.

FYI, from 1.9 the Array for RHS is not generated if it is not needed.

not generated

a, b = c, d
a, b, c = d, e

generated

a = b, c
a = b, c, d

I remain the spec because of compatibility.
#=> go to matz issue.

As you say the case which need to generate an array are not major case,
it is not performance problem in my opinion.

Thanks,
Koichi

--
// SASADA Koichi at atdot dot net

#3 - 10/15/2012 04:55 AM - headius (Charles Nutter)

I thought I replied to ko1, but must not have.

I believe MRI is using the same trick JRuby is...specifically, when the masgn's result is not used, it is not created. However, that does not help cases where masgn happens to be the last line in a method but its result is not used.

For example, the first case cannot optimize the masgn array away, but the second case can. The difference on even this small test is almost 2x GC runs:

system ~/projects/jruby $ ruby-2.0.0 -e "GC::Profiler.enable; class A; def initialize(a, b); @a, @b = a, b; end; end; 100000.times {A.new(1, 2)}; GC::Profiler.report"
GC 17 invokes.

Index	Invoke Time(sec)	Use Size(byte)	Total Size(byte)	Total Object	GC Time(ms)
1	0.012	187520	701760	17544	0.25600000000000100453
2	0.014	187520	701760	17544	0.23500000000000081934
3	0.017	187440	701760	17544	0.22700000000000150613
4	0.020	187440	701760	17544	0.23100000000000203704
5	0.023	187440	701760	17544	0.23700000000000109868
6	0.025	187440	701760	17544	0.22199999999999997513
7	0.028	187440	701760	17544	0.20999999999999838241
8	0.031	187440	701760	17544	0.23499999999999909850
9	0.033	187440	701760	17544	0.24400000000000116041
10	0.036	187440	701760	17544	0.22400000000000197531
11	0.039	187440	701760	17544	0.22599999999999703659
12	0.041	187440	701760	17544	0.22099999999999897504
13	0.044	187440	701760	17544	0.19999999999999878986
14	0.046	187440	701760	17544	0.20900000000000085176
15	0.049	187440	701760	17544	0.20699999999999885159
16	0.052	187440	701760	17544	0.23900000000000309885

system ~/projects/jruby $ ruby-2.0.0 -e "GC::Profiler.enable; class A; def initialize(a, b); @a, @b = a, b; nil; end; end; 100000.times {A.new(1, 2)}; GC::Profiler.report"
GC 9 invokes.

Index	Invoke Time(sec)	Use Size(byte)	Total Size(byte)	Total Object	GC Time(ms)
1	0.012	187400	701760	17544	0.26500000000000134559
2	0.017	187400	701760	17544	0.28699999999999903366
3	0.021	187360	701760	17544	0.23799999999999862932
4	0.025	187360	701760	17544	0.22500000000000297540
5	0.030	187360	701760	17544	0.20799999999999985167
6	0.034	187360	701760	17544	0.20599999999999785150
7	0.038	187360	701760	17544	0.20499999999999685141
8	0.042	187360	701760	17544	0.22099999999999897504

For a result that is used so rarely, it seems a shame to require masgn to always return an array when used as an expression.

#4 - 10/15/2012 03:29 PM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

- Assignee set to matz (Yukihiro Matsumoto)

Changing return value from massign would be agaist 2.0 compatibility policy. Maybe in 3.0.
Method inlining in JRuby would help compatibility here as well, wouldn't it?

Matz.

#5 - 10/16/2012 02:10 AM - headius (Charles Nutter)

Method inlining could help if we do it before handing off to the JVM, since we'd see that masgn result is not used...but that's still pretty far away from reality. The JVM might be able to eliminate the array allocation, but it's a very complicated operation and it will be difficult to see that it is zero-sum.