Ruby - Feature #5809

Benchmark#bm: remove the label_width parameter

12/26/2011 07:37 PM - Eregon (Benoit Daloze)

Status:	Closed	
Priority:	Normal	
Assignee:	naruse (Yui NARUSE)	
Target version:	2.6	

Description

Hello,

I would like to keep on improving the benchmark library.
Feature #4197 was mostly a clean-up, this intend to improve the existing methods.

First, I would like to make Benchmark#bm smarter.
I think than having to specify the maximum width of the labels as an argument is not natural, and probably not the best API.
An easy way to calculate the maximum width is to store all blocks and labels and run them after the #bm block is yielded.
That's the way #bmbm does it (although some people don't know #bmbm can calculate the width itself).
It would also avoid the duplication between the Report and Job classes, and make #bm more consistent with #bmbm.

There is however a good reason it is not done this way:
It lets #report return immediately the Benchmark::Tms, which can then be assigned to a variable, and be used to easily display the total, average, etc:

```
Benchmark.bm(7, ">total:", ">avg:") do |x|
tf = x.report("for:")   { for i in 1..n; a = "1"; end }
tt = x.report("times:") { n.times do   ; a = "1"; end }
[tf+tt, (tf+tt)/3]
end
```

=>

```
              user     system      total        real
```

```
for:     0.000000   0.000000   0.000000 (  0.000054)
times:   0.000000   0.000000   0.000000 (  0.000027)

      total:  0.000000   0.000000   0.000000 (  0.000081)
      avg:    0.000000   0.000000   0.000000 (  0.000027)
```

I am not sure this is worth having to give the width.
I think this feature is very rarely used (I actually never saw a code using it, except the code in the documentation). Please show me if I'm wrong.

There are some workarounds.

First, I made Benchmark#benchmark returns the times of the #reports (an Array of Tms).
One can then easily store the results of #bm and print them (however it will not indent them automatically, but that could be fixed).

```
tf,tt = Benchmark.bm { ... }
puts ">total: #{(tf+tt)}"
puts ">avg:  #{(tf+tt)/3}"
```

Second, it is actually possible to support the same feature, even when the #report blocks are executed after.
This is done by creating "delay-able"/lazy objects, which remember the calls, and make the real calls by calling #compute on them.
Benchmark#benchmark could then execute them when the #bm block is yielded, and it would be transparent to the user (except if he prints the result of #report inside the #bm block).

The second is not small to implement, and might not belong to the benchmark library, but could be useful in general. It also adds some significant complexity.

What do you think?

History

#1 - 12/27/2011 12:56 AM - trans (Thomas Sawyer)

All makes very good sense. But I would avoid complexity. Sometimes things need to change, people have to adjust. I'm not sure I ever even knew that #report returned the time, and I imagine you are right, it was rarely used, so I don't think it is too much to ask.

OTOH, maybe just let people worry about spacing themselves.

```
Benchmark.bm do |x|
   tf = x.report("for:   ") { for i in 1..n; a = "1"; end }
 tt = x.report("times: ") { n.times do   ; a = "1"; end }
end
```

Or let #report take an optional Fixnum argument.

```
Benchmark.bm do |x|
   tf = x.report("for:", 7)   { for i in 1..n; a = "1"; end }
 tt = x.report("times:", 7) { n.times do   ; a = "1"; end }
end
```

Also, the width could be assigned:

```
Benchmark.bm do |x|
x.width = 7
   tf = x.report("for:")   { for i in 1..n; a = "1"; end }
 tt = x.report("times:") { n.times do   ; a = "1"; end }
end
```

But I also don't like the "header -> array mapping" thing --that's a pretty fugly API. I'd much rather do something like:

```
Benchmark.bm do |x|
   tf = x.report("for:   ") { for i in 1..n; a = "1"; end }
 tt = x.report("times: ") { n.times do   ; a = "1"; end }
x.report("total: ", "#{tf + tt}")
x.report("avg:   ", "#{(tf+tt)/3}")
end
```

#2 - 12/27/2011 07:15 AM - Eregon (Benoit Daloze)

Thomas Sawyer wrote:

> All makes very good sense. But I would avoid complexity. Sometimes things need to change, people have to adjust. I'm not sure I ever even knew that #report returned the time, and I imagine you are right, it was rarely used, so I don't think it is too much to ask.

Thanks, I believe dropping the unused feature is the way to go, but I want to know other's opinions.

> OTOH, maybe just let people worry about spacing themselves.

I think complicated formatters should not be part of the library, but aligning the times make them easier to read and still easy to parse (and the implementation is small). I especially want to avoid the user to have to think about spaces when making a benchmark.

> But I also don't like the "header -> array mapping" thing --that's a pretty fugly API. I'd much rather do something like:
>
> ```
> Benchmark.bm do |x|
>    tf = x.report("for:   ") { for i in 1..n; a = "1"; end }
>  tt = x.report("times: ") { n.times do   ; a = "1"; end }
> x.report("total: ", "#{tf + tt}")
> x.report("avg:   ", "#{(tf+tt)/3}")
> end
> ```

Yeah, it's a rather special API. Thanks for the code, it gave me some ideas to implement this kind of feature.

#3 - 12/28/2011 07:32 PM - naruse (Yui NARUSE)

If your suggestion is to provide easy way to pretty enough benchmark method, how about following:

```
IMPLEMENTATION:
def Benchmark(tbl)
len = tbl.keys.map(&:size).max + 1
```

```
total = ">total:"
avg   = ">avg:"
len = [len, total.length, avg.length].max
Benchmark.bm(len, total, avg) do |x|
sum = nil
tbl.each do |k, v|
r = x.report(k + ':', &v)
sum = sum ? sum + r : r
end
[sum, sum/tbl.size]
end
end
```

EXAMPLE:

```
n=1_000_000
Benchmark(
"for"   => ->{ for i in 1..n; a = "1"; end},
"times" => ->{ n.times do   ; a = "1"; end},
)
```

OUTPUT:

```
user     system      total        real
for:     0.367188   0.000000   0.367188 (  0.371534)
times:    0.359375   0.000000   0.359375 (  0.357455)

      total:   0.726562   0.000000   0.726562 (  0.728989)
      avg:     0.363281   0.000000   0.363281 (  0.364494)
```

#4 - 01/03/2012 04:59 AM - Eregon (Benoit Daloze)

Yui NARUSE wrote:

> If your suggestion is to provide easy way to pretty enough benchmark
> method, how about following:

Nice! That could be a new and concise way to do benchmarks (although it might be less flexible).

My suggestion is to improve Benchmark#benchmark (and so #bm) by removing the label_width argument, which I think is unnecessary (and so calculate it).

I would keep compatibility (use the width if given), but remove the feature of returning the time directly at #report, which is not done for #bmbm anyway.

Do you think it is fine to remove that (as far as I know) unused feature? Or should I absolutely keep it (although it is inconsistent and duplicate the code) ?

#5 - 01/05/2012 12:13 PM - naruse (Yui NARUSE)

Benoit Daloze wrote:

> My suggestion is to improve Benchmark#benchmark (and so #bm) by removing
> the label_width argument, which I think is unnecessary (and so calculate
> it).
>
> I would keep compatibility (use the width if given), but remove the feature
> of returning the time directly at #report, which is not done for #bmbm
> anyway.
>
> Do you think it is fine to remove that (as far as I know) unused feature?
> Or should I absolutely keep it (although it is inconsistent and duplicate
> the code) ?

Breaking compatibility makes the change hard to accept even if it seems not be used.
Keep and deprecate old method, and create new method is better.

#6 - 03/29/2012 01:44 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to naruse (Yui NARUSE)

Naruse-san, what does this ticket need? A better new name?
Could you please create a patch?

--
Yusuke Endoh mame@tsg.ne.jp

#7 - 03/29/2012 09:46 PM - Eregon (Benoit Daloze)

@naruse (Yui NARUSE): Sorry, I meant to answer to your reply much earlier, I just didn't make my mind at that time.

I understand the compatibility need, so I'll try to create a new method and refactor without changing existing API.

So my goal is both to add a convenient method for formatting, using as much as possible what is already available, and refactor the existing code to avoid duplication where possible.

Also, the example you gave is certainly worth investigating!

I'll get back to this as soon as I can.

#8 - 03/30/2012 05:32 PM - naruse (Yui NARUSE)

- Status changed from Assigned to Feedback

#9 - 11/20/2012 10:19 PM - mame (Yusuke Endoh)

- Target version set to 2.6

#10 - 11/08/2014 08:53 PM - Eregon (Benoit Daloze)

- Status changed from Feedback to Closed

Closing this since it would significantly break compatibility.