Ruby Benchmark

Ne faites pas confiance à votre instinct

KissKiss
BankBank®.com
Maison de Créativité

Problématique

```
hash = {
 42 => { min: 5, max: 10 },
 43 => { min: 4, max: 9 },
 44 => { min: 3, max: 7 },
 # …
}
```

Je souhaite faire la somme des `min`.

Solutions

```ruby
require 'active_support/core_ext/enumerable'

hash = {
 42 => { min: 5, max: 10 },
 43 => { min: 4, max: 9 },
 44 => { min: 3, max: 7 },
}

hash.values.inject(0) { |inc, h| inc + h[:min] }
hash.sum              { |_, h| h[:min] }
```

Benchmark !

```ruby
require 'benchmark'
Benchmark.bm do |x|
  x.report { 500.times { … } }
  x.report { 500.times { … } }
end
```

Quelle est la solution la plus performante ?

```
each_inject:       hash.each_value.inject(0) { |i, h| i + h[:min] }
each_map_reduce:   hash.each_value.map       { |h| h[:min] }.reduce(:+)
each_sum:          hash.each_value.sum       { |v| v[:min] }
inject:            hash.inject(0)            { |i, h| i + h[1][:min] }
inject_block_var:  hash.inject(0)            { |i, (_, v)| i + v[:min] }
sum_block_var:     hash.sum                  { |_, v| v[:min] }
values_inject:     hash.values.inject(0)     { |i, h| i + h[:min] }
values_map_reduce: hash.values.map           { |h| h[:min] }.reduce(:+)
values_sum:        hash.values.sum           { |v| v[:min] }
```

Benchmark !

```ruby
require 'bmark' # https://gist.github.com/sunny/c47982974f749da82b6f
require 'active_support/core_ext/enumerable'

hash = {}
100.times do |i|
  hash[i] = { min: 5, max: 10 }
end

bmark 200_000,
  each_inject:       -> { hash.each_value.inject(0) { |i, h| i + h[:min] } },
  each_map_reduce:   -> { hash.each_value.map       { |h| h[:min] }.reduce(:+) },
  each_sum:          -> { hash.each_value.sum       { |v| v[:min] } },
  inject:            -> { hash.inject(0)            { |i, h| i + h[1][:min] } },
  inject_block_var:  -> { hash.inject(0)            { |i, (_, v)| i + v[:min] } },
  sum_block_var:     -> { hash.sum                  { |_, v| v[:min] } },
  values_inject:     -> { hash.values.inject(0)     { |i, h| i + h[:min] } },
  values_map_reduce: -> { hash.values.map           { |h| h[:min] }.reduce(:+) },
  values_sum:        -> { hash.values.sum           { |v| v[:min] } }
```

Roulements de tambours…

Résultats

```
                      user     system      total        real
each_inject       2.850000   0.000000   2.850000 (  2.858093)
each_map_reduce   4.290000   0.000000   4.290000 (  4.296114)
each_sum          5.120000   0.010000   5.130000 (  5.126060)
inject            4.020000   0.000000   4.020000 (  4.031549)
inject_block_var  4.210000   0.010000   4.220000 (  4.213347)
sum_block_var     6.260000   0.000000   6.260000 (  6.263503)
values_inject     2.290000   0.010000   2.300000 (  2.296429)
values_map_reduce 3.170000   0.010000   3.180000 (  3.188121)
values_sum        3.960000   0.010000   3.970000 (  3.966809)
```

Résultats

```
each_inject:        hash.each_value.inject(0) { |i, h| i + h[:min] }      # 2.86
each_map_reduce:    hash.each_value.map       { |h| h[:min] }.reduce(:+)  # 4.23
each_sum:           hash.each_value.sum       { |v| v[:min] }             # 5.13
inject:             hash.inject(0)            { |i, h| i + h[1][:min] }   # 4.03
inject_block_var:   hash.inject(0)            { |i, (_, v)| i + v[:min] } # 4.21
sum_block_var:      hash.sum                  { |_, v| v[:min] }          # 6.26
values_inject:      hash.values.inject(0)     { |i, h| i + h[:min] }      # 2.30
values_map_reduce:  hash.values.map           { |h| h[:min] }.reduce(:+)  # 3.19
values_sum:         hash.values.sum           { |v| v[:min] }             # 3.97
```

Merci !