mruby と CRuby の性能比較

2014/12/30
OSS推進フォーラム アプリケーション部会 冨田 昌宏

目的

mruby を機器組み込みではなく、Linux 上のアプリケーションに組み込む言語として使用、または CRuby の代替として使用する際の特性を知るために、CRuby との性能を比較する。

バージョン

mruby
: IIJ mruby をベースにモジュールを追加したもの。詳細は後述。

CRuby
: ruby 2.2.0p0 (2014-12-25 revision 49005) [x86_64-linux]

測定環境

PC
: ThinkPad X220

CPU
: Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz

Memory
: 8GB

OS
: Ubuntu 14.10 (Kernel 3.16.0-28)

測定方法

The Ruby Benchmark Suite https://github.com/acangiano/ruby-benchmark-suite.git のベンチマークを使用して、時間とメモリ使用量を計測する。

- macro-benchmarks と micro-benchmarks の中の mruby で動作するもののみを対象とした。
- mruby でタイムアウトするテストがあったためタイムアウト値を1000秒にした。
- mruby でメモリを食い潰してしまうテストがあったため、メモリ空間を 4GB に制限して実行した。(`ulimit -S -v 4194304`)
- いくつかのテストを mruby で動作するように改変して実行した。
- メモリ使用量は、オリジナルはベンチマーク終了時の VmRSS(実メモリ使用量)を計測していたが、プログラムやライブラリの大きさを除いたメモリ使用量を測定するため、VmData(ヒープ領域のメモリ量)を計測するようにした。

mruby

```
% rake bench:dir DIR=benchmarks/m\?cro-benchmarks TIMEOUT=1000 VM=mruby
```

CRuby

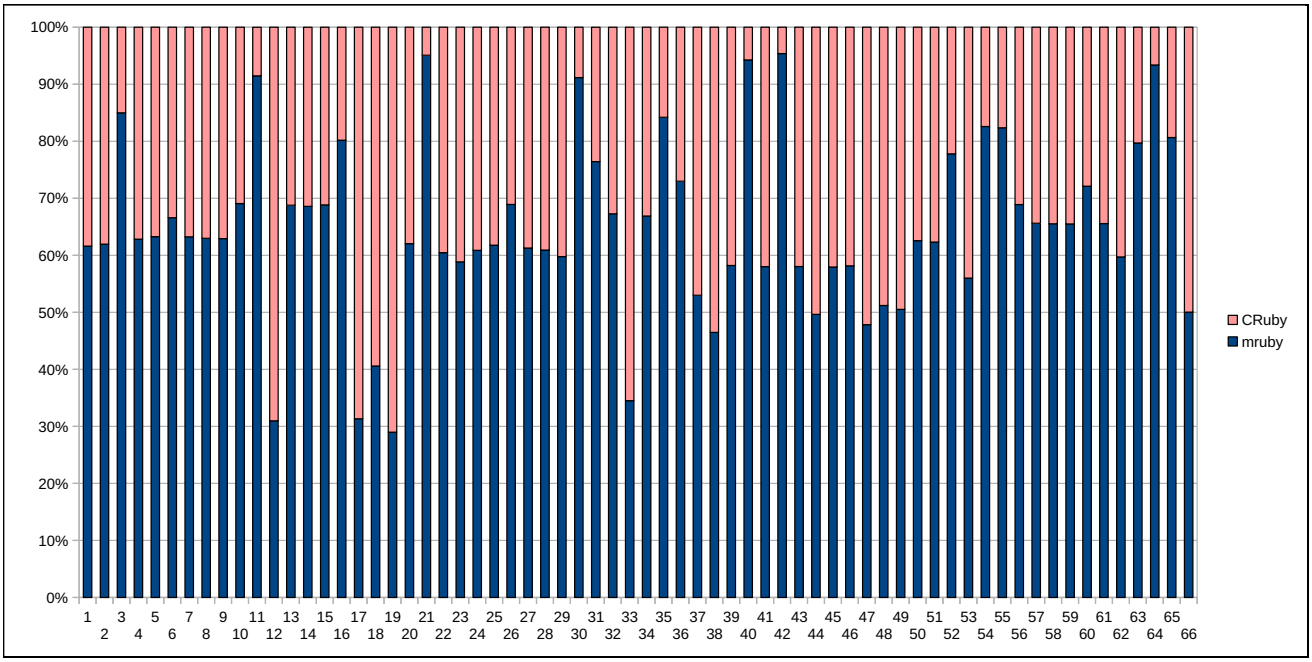
```
% rake bench:dir DIR=benchmarks/m\?cro-benchmarks TIMEOUT=1000 VM=ruby
```

結果

No.	name	parameter	mruby time(sec)	CRuby time(sec)	mruby memory(KB)	CRuby memory(KB)
1	bm_app_fib.rb	30	0.32105	0.19999	1,484	5,728
2	bm_app_fib.rb	35	3.55515	2.18296	1,484	5,728
3	bm_app_pentomino.rb	1	139.41174	24.62176	2,933	5,963
4	bm_app_tak.rb	7	0.26792	0.15845	1,484	5,728
5	bm_app_tak.rb	8	0.77943	0.45240	1,484	5,728
6	bm_app_tak.rb	9	2.34484	1.17617	1,484	5,728
7	bm_app_tarai.rb	3	0.92912	0.53995	1,484	5,704
8	bm_app_tarai.rb	4	1.11567	0.65568	1,484	5,704
9	bm_app_tarai.rb	5	1.34884	0.79435	1,484	5,704
10	bm_binary_trees.rb	1	23.70115	10.60833	293,854	38,723
11	bm_dirp.rb	10000	13.94624	1.29917	2,310	5,644
12	bm_eval.rb	1000000	3.28600	7.32403	2,172	7,064
13	bm_fannkuch.rb	6	0.00312	0.00142	1,560	5,704
14	bm_fannkuch.rb	8	0.25645	0.11746	1,502	5,704
15	bm_fannkuch.rb	10	33.60273	15.20352	2,333	5,835
16	bm_fasta.rb	1000000	44.18359	10.91848	25,420	12,610
17	bm_fiber_ring.rb	10	0.00012	0.00027	1,564	19,506
18	bm_fiber_ring.rb	100	0.00666	0.00975	1,564	123,514
19	bm_fiber_ring.rb	1000	0.66968	1.64103	2,940	1,089,694
20	bm_fractal.rb	5	1.30513	0.79814	1,500	5,728
21	bm_gc_array.rb	1	180.42288	9.31099	4,185,877	240,055
22	bm_gc_mb.rb	500000	0.21094	0.13788	47,590	53,566
23	bm_gc_mb.rb	1000000	0.41138	0.28763	91,302	106,339
24	bm_gc_mb.rb	3000000	1.31169	0.84314	335,746	299,958
25	bm_gc_string.rb	1	4.87376	3.01516	175,570	153,377
26	bm_knucleotide.rb	1	1.32183	0.59595	41,296	38,382
27	bm_list.rb	1000	0.02918	0.01844	27,717	25,466
28	bm_list.rb	10000	1.12071	0.71900	2,000,180	96,751
29	bm_mandelbrot.rb	1	15.57870	10.48232	1,302	5,704
30	bm_mergesort.rb	1	5.91457	0.57263	94,240	22,179
31	bm_mergesort_hongli.rb	3000	2.93958	0.90599	3,405	6,424
32	bm_monte_carlo_pi.rb	10000000	6.75601	3.28582	1,484	5,704
33	bm_mpart.rb	300	0.09070	0.17215	1,484	5,900
34	bm_nbody.rb	100000	2.58133	1.27773	1,576	5,704
35	bm_norvig_spelling.rb	50	26.19799	4.91191	154,028	21,366
36	bm_nsieve.rb	9	5.45480	2.01764	512,597	115,724
37	bm_open_many_files.rb	50000	0.24298	0.21545	1,484	6,492
38	bm_parse_log.rb	100	0.53858	0.62022	2,699	10,844
39	bm_partial_sums.rb	2500000	4.96196	3.56126	1,564	5,648
40	bm_quicksort.rb	1	21.45094	1.30423	90,979	40,854
41	bm_rcs.rb	100	0.27599	0.19966	1,488	5,768
42	bm_read_large.rb	100	40.48686	1.95911	4,353	5,644
43	bm_regex_dna.rb	20	3.23953	2.34210	10,394	14,788

No.	name	parameter	mruby time(sec)	CRuby time(sec)	mruby memory(KB)	CRuby memory(KB)
44	bm_simple_connect.rb	1	0.00016	0.00017	1,484	5,900
45	bm_simple_connect.rb	100	0.00752	0.00546	1,484	5,900
46	bm_simple_connect.rb	500	0.03727	0.02682	1,484	5,900
47	bm_simple_server.rb	1	0.00017	0.00019	1,484	5,912
48	bm_simple_server.rb	100	0.00133	0.00127	1,484	5,912
49	bm_simple_server.rb	100000	1.19364	1.16892	1,484	12,931
50	bm_so_ackermann.rb	7	0.10774	0.06445	1,636	5,644
51	bm_so_ackermann.rb	9	1.71398	1.03619	2,476	5,644
52	bm_so_array.rb	9000	4.58950	1.30979	1,741	5,868
53	bm_so_exception.rb	500000	1.45366	1.14175	1,420	10,698
54	bm_so_lists.rb	1000	14.32586	3.02005	145,779	97,580
55	bm_so_lists_small.rb	1000	2.88095	0.61655	17,798	51,722
56	bm_so_matrix.rb	60	0.93273	0.42090	2,542	5,648
57	bm_so_object.rb	500000	0.62995	0.32999	1,529	5,728
58	bm_so_object.rb	1000000	1.25856	0.66194	1,652	5,728
59	bm_so_object.rb	1500000	1.88662	0.99376	1,784	5,728
60	bm_so_sieve.rb	4000	15.48247	5.98386	452,668	47,331
61	bm_spectral_norm.rb	100	0.22333	0.11725	1,560	5,704
62	bm_string_concat.rb	10000000	2.55351	1.72220	1,056,174	1,437,694
63	bm_sudoku.rb	1	8.06493	2.05373	2,854	5,644
64	bm_sum_file.rb	100	43.55694	3.08530	4,350	5,704
65	bm_word_anagrams.rb	1	10.71589	2.56914	118,618	54,749
66	bm_write_large.rb	100	2.22511	2.22108	1,560	5,644

時間の比較



全般的に CRuby の方が mruby よりも高速。

mruby の方が高速なもの上位5件:

No.	name	parameter	mruby time(sec)	CRuby time(sec)	mruby memory(KB)	CRuby memory(KB)
19	bm_fiber_ring.rb	1000	0.66968	1.64103	2,940	1,089,694
12	bm_eval.rb	1000000	3.28600	7.32403	2,172	7,064
17	bm_fiber_ring.rb	10	0.00012	0.00027	1,564	19,506
33	bm_mpart.rb	300	0.09070	0.17215	1,484	5,900
18	bm_fiber_ring.rb	100	0.00666	0.00975	1,564	123,514

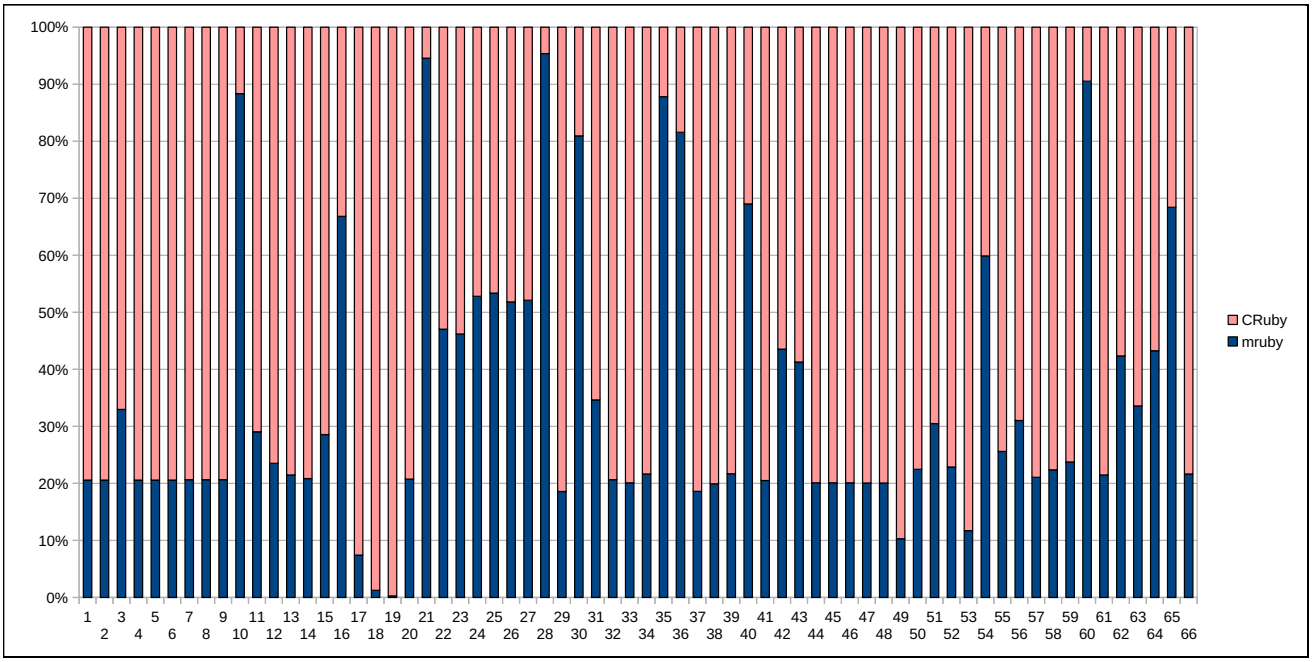
- bm_fiber_ring.rb : Fiber
- bm_eval.rb : 文字列の eval
- bm_mpart.rb : 7行のファイルを3行ずつに分割したファイルを作成する

CRuby の方が高速なもの上位5件:

No.	name	parameter	mruby time(sec)	CRuby time(sec)	mruby memory(KB)	CRuby memory(KB)
42	bm_read_large.rb	100	40.48686	1.95911	4,353	5,644
21	bm_gc_array.rb	1	180.42288	9.31099	4,185,877	240,055
40	bm_quicksort.rb	1	21.45094	1.30423	90,979	40,854
64	bm_sum_file.rb	100	43.55694	3.08530	4,350	5,704
11	bm_dirp.rb	10000	13.94624	1.29917	2,310	5,644

- bm_read_large.rb : 100000行のファイルを1行ずつ読み込み
- bm_gc_array.rb : 100要素の Array オブジェクトを100万個作成
- bm_quicksort.rb : 100000件のクイックソート
- bm_sum_file.rb : 100000行のファイルに書かれた数値を合計する
- bm_dirp.rb : ディレクトリ中のファイル一覧

メモリ消費量の比較



全般的に mruby の方がメモリ消費量が少ない。

mruby の方がメモリ消費量が少ないもの上位5件:

No.	name	parameter	mruby time(sec)	CRuby time(sec)	mruby memory(KB)	CRuby memory(KB)
19	bm_fiber_ring.rb	1000	0.66968	1.64103	2,940	1,089,694
18	bm_fiber_ring.rb	100	0.00666	0.00975	1,564	123,514
17	bm_fiber_ring.rb	10	0.00012	0.00027	1,564	19,506
49	bm_simple_server.rb	100000	1.19364	1.16892	1,484	12,931
53	bm_so_exception.rb	500000	1.45366	1.14175	1,420	10,698

- bm_fiber_ring.rb : Fiber
- bm_simple_server.rb : TCPソケットの読み書き
- bm_so_exception.rb : 例外

CRuby の方がメモリ消費量が少ないもの上位5件:

No.	name	parameter	mruby time(sec)	CRuby time(sec)	mruby memory(KB)	CRuby memory(KB)
28	bm_list.rb	10000	1.12071	0.71900	2,000,180	96,751
21	bm_gc_array.rb	1	180.42288	9.31099	4,185,877	240,055
60	bm_so_sieve.rb	4000	15.48247	5.98386	452,668	47,331
10	bm_binary_trees.rb	1	23.70115	10.60833	293,854	38,723
35	bm_norvig_spelling.rb	50	26.19799	4.91191	154,028	21,366

- bm_list.rb : リンクリスト
- bm_gc_array.rb : 100要素の Array オブジェクトを100万個作成
- bm_so_sieve.rb : 8192 以下の素数の数
- bm_binary_trees.rb : バイナリツリー
- bm_norvig_spelling.rb : スペルチェッカ

考察

1. mruby は Fiber が高速。次のスクリプトで試しても同様の傾向が見られた。

```
t = Time.now
fiber = Fiber.new do
  while true
    Fiber.yield
  end
end
1000000.times do
  fiber.resume
end
p Time.now - t
```

```
% mruby ./fiber.rb
0.3379620
% ruby ./fiber.rb
0.620244294
```

2. mruby は eval が高速。次のスクリプトで試しても同様の傾向が見られた。

```
s = '1'
t = Time.now
100000.times do |i|
  eval s
end
p Time.now - t
```

```
% mruby ./eval.rb
0.2680730
% ruby ./eval.rb
0.525556645
```

3. mruby で高速に Array オブジェクトの生成&破棄を行う場合、メモリ使用量が膨れる。

次のスクリプトで検証。

```
n = ARGV[0].to_i
gc = ARGV[1]
t = Time.now
n.times do
  100000.times.map do
    Array.new(100)
  end
  GC.start if gc
end
p Time.now - t
system "grep VmPeak /proc/#{$$}/status"
```

```
# 繰り返し回数 1 & GCなし
% mruby array.rb 1
1.6690380
VmPeak:       236536 kB

# 繰り返し回数 5 & GCなし
% mruby array.rb 5
8.2672190
VmPeak:      1151004 kB

# 繰り返し回数10 & GCなし
% mruby array.rb 10
16.52840
```

```
VmPeak:      2294176 kB

# 繰り返し回数 5 & GCあり
% mruby array.rb 5 gc
8.7810810
VmPeak:       237236 kB

# 繰り返し回数 10 & GCあり
% mruby array.rb 10 gc
17.1099120
VmPeak:       237236 kB
```

繰り返し数に比例してメモリ使用量が増えるが、明に `GC.start` を行うとちゃんとメモリが解放される。オブジェクトの生成速度に GC が追いついていないためではないかと思われる。

4. mruby はファイル読み込みが遅い

iij/mruby-io では `IO#read`, `IO#readline` は C ではなく Ruby で実装されているためだと思われる。

mruby

IIJ mruby https://github.com/iij/mruby.git commit:6f4b2791 (2014/12/28 07:02:38) をベースに次の変更を行った。

mrbgems 配下のモジュールを有効化。ただし次のものは除く。

- bin-debugger
    - CFLAGS=-O3 でコンパイルできなかったため。
- string-utf8
    - 文字列処理が非常に遅くなるため。

機能追加のため次のIIJモジュールを有効化。

- mruby-dir
- mruby-errno
- mruby-io
- mruby-pack
- mruby-process
- mruby-require
- mruby-socket
- mruby-tempfile

機能追加のため次の外部モジュールを有効化。

- https://github.com/mattn/mruby-onig-regexp.git
- https://github.com/ksss/mruby-file-stat.git
- https://github.com/gromnitsky/mruby-dir-glob.git

変更内容の詳細は https://github.com/tmtm/mruby/tree/benchmark 参照。

また、今回の測定中に以下のバグを発見して報告した(いずれも修正済み)。

- mruby/mruby #2626 `Array#[float, int]` が ArgumentError になる
- mruby/mruby #2649 `String#slice!` が TypeError または不正な値になる
- mruby/mruby #2650 `String#[float]` が不正な値を返す
- mruby/mruby #2655 `Numeric#step` が無限ループになる
- iij/mruby-io #31 `IO#sysread` が SEGV になる
- iij/mruby-io #35 `IO#read` が大量のオブジェクトを生成する

コンパイル方法:

```
% CFLAGS=-O3 make
```

ruby-benchmark-suite

mruby で動作するように以下の変更を行った。

- 整数除算の結果が整数になるように `x / y` を `x.div y` に変更。
- printf の第一引数が IO の場合に対応していないため、`printf(io,...)` を `io.printf(...)` に変更。
- `Dir[pattern]` が動作しないため `Dir.glob(pattern)` に変更。
- 正規表現リテラル中に正規表現オブジェクトを埋め込めないため、書き換え。
- `Numeric#zero?` がないため、`== 0` に変更。
- `Range#step` がないため、`Integer#step` に変更。

mruby では動作しないので無効化したもの。

- zlib, stringio ライブラリがないため。
    - `macro-benchmarks/bm_gzip.rb`

- complex ライブラリがないため。
  - `micro-benchmarks/bm_app_mandelbrot.rb`
- date ライブラリがないため。
  - `micro-benchmarks/bm_cal.rb`
- ffi ライブラリがないため。
  - `micro-benchmarks/bm_ffi_printf.rb`
- pathname ライブラリがないため。
  - `micro-benchmarks/bm_pathname.rb`
- mathn ライブラリがないため。
  - `micro-benchmarks/bm_primes.rb`
- Matrix クラスがないため。
  - `macro-benchmarks/bm_hilbert_matrix.rb`
- benchmark ライブラリがないため。
  - `micro-benchmarks/bm_str_chars.rb`
- 整数演算がオーバーフローし、浮動小数点になってしまうため。
  - `micro-benchmarks/bm_app_factorial.rb`
  - `micro-benchmarks/bm_lucas_lehmer.rb`
  - `micro-benchmarks/bm_pi.rb`
- Thread クラスがないため。
  - `micro-benchmarks/bm_count_multithreaded.rb`
  - `micro-benchmarks/bm_count_shared_thread.rb`
  - `micro-benchmarks/bm_mbari_bogus1.rb`
  - `micro-benchmarks/bm_mbari_bogus2.rb`
  - `micro-benchmarks/bm_observ.rb`
  - `micro-benchmarks/bm_socket_transfer_1mb.rb`
  - `micro-benchmarks/bm_socket_transfer_1mb_noblock.rb`
- `String#tr` メソッドがないため。
  - `micro-benchmarks/bm_reverse_complement.rb`
- `Integer#[]` メソッドがないため。
  - `micro-benchmarks/bm_meteor_contest.rb`
- `String#[]` が数値を返すことを期待しているため。※ CRuby でも 1.9 以降では動作しない
  - `micro-benchmarks/bm_nsieve_bits.rb`
- `String#count` メソッドがないため。
  - `micro-benchmarks/bm_so_count_words.rb`

変更内容の詳細は https://github.com/tmtm/ruby-benchmark-suite/tree/for_mruby 参照。