Ruby master - Misc #15802

Reduce the minimum string buffer size from 127 to 63 bytes

04/27/2019 04:40 PM - methodmissing (Lourens Naudé)

Status:	Open	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	

Description

References Github PR https://github.com/ruby/ruby/pull/2151 - another small change, but posting here for further discussion.

While having a look through String specific allocation paths with dhat on redmine I noticed many string buffer use cases actually need way less than the current 128 byte minimum size (127 current minimum size + sentinel).

These auxiliary buffers are malloc heap specific as the String type is not supported by the transient heap due to complexity.

How to interpret the DHAT output

From the example output below, we can draw the following conclusions for the specific allocation site leading up to rb_str_buf_new:

- Total allocated size of 434944 bytes with a very low read and write access ratio under 25%
- The buffer is thus 75% larger than it should be for this particular site (more examples further down below)
- Short lived as expected from a buffer use case, but did occupy non-insignificant heap space still for a fair amount of time
- The 0s at the tail end of the output represent memory never accessed (rows are byte offsets)
- As an aside, this particular string's first few characters are hot (accessed more frequently than the tail end)

```
==26579== -------------------- 95 of 300 --------------------
==26579== max-live:    330,368 in 2,581 blocks
==26579== tot-alloc:   434,944 in 3,398 blocks (avg size 128.00)
==26579== deaths:      3,347, at avg age 368,102,626 (2.64% of prog lifetime)
==26579== acc-ratios:  0.22 rd, 0.25 wr  (97,275 b-read, 110,341 b-written)
==26579==    at 0x4C2DECF: malloc (in /usr/lib/valgrind/vgpreload_exp-dhat-amd64-linux.so)
==26579==    by 0x1521D3: objspace_xmalloc0 (gc.c:9407)
==26579==    by 0x284A9B: rb_str_buf_new (string.c:1331)
==26579==    by 0x31B9F7: rb_ary_join (array.c:2331)
==26579==    by 0x2E5B4E: vm_call_cfunc_with_frame (vm_insnhelper.c:2207)
==26579==    by 0x2E5B4E: vm_call_cfunc (vm_insnhelper.c:2225)
==26579==    by 0x2F7C7C: vm_sendish (vm_insnhelper.c:3623)
==26579==    by 0x2F7C7C: vm_exec_core (insns.def:789)
==26579==    by 0x2EE34D: rb_vm_exec (vm.c:1892)
==26579==    by 0x2EEEED: invoke_iseq_block_from_c (vm.c:1104)
==26579==    by 0x2EEEED: invoke_block_from_c_bh (vm.c:1122)
==26579==    by 0x2EEEED: vm_yield (vm.c:1167)
==26579==    by 0x2EEEED: rb_yield_0 (vm_eval.c:980)
==26579==    by 0x2EEEED: rb_yield_1 (vm_eval.c:986)
==26579==    by 0x2EEEED: rb_yield (vm_eval.c:996)
==26579==    by 0x31153B: rb_ary_each (array.c:2087)
==26579==    by 0x2E5B4E: vm_call_cfunc_with_frame (vm_insnhelper.c:2207)
==26579==    by 0x2E5B4E: vm_call_cfunc (vm_insnhelper.c:2225)
==26579==    by 0x2F7D2B: vm_sendish (vm_insnhelper.c:3623)
==26579==    by 0x2F7D2B: vm_exec_core (insns.def:771)
==26579==    by 0x2EE34D: rb_vm_exec (vm.c:1892)
==26579==
==26579== Aggregated access counts by offset:
==26579==
==26579== [   0]  11407 8731 9660 11112 11171 11724 13165 13609 10826 10998 11436 11420 11405 1029
5 9710 9316
==26579== [  16]  5664 4959 3874 3607 2965 2395 1908 1509 1285 967 597 261 149 141 140 124
==26579== [  32]  73 57 56 55 55 55 54 52 51 51 51 51 51 51 51 51
==26579== [  48]  34 34 34 34 34 34 17 0 0 0 0 0 0 0 0 0
==26579== [  64]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [  80]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [  96]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [ 112]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Why lower is better

The easiest reproducible case from the benchmark suite is the require benchmark (the allocation site below is from Rails though - DHAT is very slow and doesn't make sense in context of a benchmark):

```
==28383== -------------------- 572 of 600 --------------------
==28383== max-live:    36,992 in 289 blocks
==28383== tot-alloc:   91,520 in 715 blocks (avg size 128.00)
==28383== deaths:      553, at avg age 908,212,488 (8.68% of prog lifetime)
==28383== acc-ratios:  6.15 rd, 0.41 wr  (563,074 b-read, 37,525 b-written)
==28383==    at 0x4C2DECF: malloc (in /usr/lib/valgrind/vgpreload_exp-dhat-amd64-linux.so)
==28383==    by 0x1521D3: objspace_xmalloc0 (gc.c:9407)
==28383==    by 0x284A9B: rb_str_buf_new (string.c:1331)
==28383==    by 0x290D56: str_gsub (string.c:5163)
==28383==    by 0x2E5B4E: vm_call_cfunc_with_frame (vm_insnhelper.c:2207)
==28383==    by 0x2E5B4E: vm_call_cfunc (vm_insnhelper.c:2225)
==28383==    by 0x2F7C7C: vm_sendish (vm_insnhelper.c:3623)
==28383==    by 0x2F7C7C: vm_exec_core (insns.def:789)
==28383==    by 0x2EE34D: rb_vm_exec (vm.c:1892)
==28383==    by 0x18B336: rb_load_internal0 (load.c:612)
==28383==    by 0x18E1B0: rb_require_internal (load.c:1028)
==28383==    by 0x18E3B2: rb_require_safe (load.c:1074)
==28383==    by 0x18E3B2: rb_f_require (load.c:821)
==28383==    by 0x2E5B4E: vm_call_cfunc_with_frame (vm_insnhelper.c:2207)
==28383==    by 0x2E5B4E: vm_call_cfunc (vm_insnhelper.c:2225)
==28383==    by 0x2F1F42: vm_call_method (vm_insnhelper.c:2712)
==28383==
==28383== Aggregated access counts by offset:
==28383==
==28383== [   0]  17936 15574 14945 15051 14538 15487 15599 15077 14658 14483 14826 14849 15232 15
238 15522 15310
==28383== [  16]  15055 15020 15143 15156 15271 14807 14656 14271 14051 13761 13365 13156 12756 12
530 12302 12002
==28383== [  32]  7652 7427 7106 6807 6594 6315 6044 5932 5750 5606 5520 5439 5347 5237 5174 5095
==28383== [  48]  2252 2211 2158 2128 2117 2088 2075 2045 2034 2018 2006 1992 1974 1973 1964 1964
==28383== [  64]  183 183 183 183 183 183 183 183 183 183 183 183 183 183 183 183
==28383== [  80]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28383== [  96]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28383== [ 112]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

lourens@CarbonX1:~/src/ruby/ruby$ /usr/local/bin/ruby --disable=gems -rrubygems -I./benchmark/lib
./benchmark/benchmark-driver/exe/benchmark-driver            --executables="compare-ruby::~/src/r
uby/trunk/ruby --disable=gems -I.ext/common --disable-gem"             --executables="built-ruby::
./miniruby -I./lib -I. -I.ext/common  -r./prelude --disable-gem" -v --repeat-count=24 -r ips $(ls
./benchmark/*require.{yml,rb} 2>/dev/null)
compare-ruby: ruby 2.7.0dev (2019-04-25 trunk 9bfc185a0d) [x86_64-linux]
built-ruby: ruby 2.7.0dev (2019-04-25 lower-str-buf-.. 9bfc185a0d) [x86_64-linux]
Calculating -------------------------------------
                     compare-ruby  built-ruby
             require        1.870       2.408 i/s -       1.000 times in 0.534865s 0.415268s

Comparison:
                           require
          built-ruby:         2.4 i/s
        compare-ruby:         1.9 i/s - 1.29x  slower

lourens@CarbonX1:~/src/ruby/ruby$ /usr/local/bin/ruby --disable=gems -rrubygems -I./benchmark/lib
./benchmark/benchmark-driver/exe/benchmark-driver            --executables="compare-ruby::~/src/r
uby/trunk/ruby --disable=gems -I.ext/common --disable-gem"             --executables="built-ruby::
./miniruby -I./lib -I. -I.ext/common  -r./prelude --disable-gem" -v --repeat-count=24 -r memory $(
ls ./benchmark/*require.{yml,rb} 2>/dev/null)
compare-ruby: ruby 2.7.0dev (2019-04-25 trunk 9bfc185a0d) [x86_64-linux]
built-ruby: ruby 2.7.0dev (2019-04-25 lower-str-buf-.. 9bfc185a0d) [x86_64-linux]
Calculating -------------------------------------
                     compare-ruby  built-ruby
             require      28.128M     26.932M bytes -       1.000 times
```

```
Comparison:
                         require
          built-ruby:  26932000.0 bytes
        compare-ruby:  28128000.0 bytes - 1.04x  larger
```

Also the hash_aref_dsym_long benchmark has significant memory reduction:

```
lourens@CarbonX1:~/src/ruby/ruby$ /usr/local/bin/ruby --disable=gems -rrubygems -I./benchmark/lib
./benchmark/benchmark-driver/exe/benchmark-driver            --executables="compare-ruby::~/src/r
uby/trunk/ruby --disable=gems -I.ext/common --disable-gem"            --executables="built-ruby::
./miniruby -I./lib -I. -I.ext/common  -r./prelude --disable-gem" -v --repeat-count=6 -r memory $(l
s ./benchmark/hash_aref_dsym_long.{yml,rb} 2>/dev/null)
compare-ruby: ruby 2.7.0dev (2019-04-26 trunk 5689c46457) [x86_64-linux]
built-ruby: ruby 2.7.0dev (2019-04-26 lower-str-buf-.. 9abd605533) [x86_64-linux]
last_commit=Reduce the minimum string buffer size from 127 to 63 bytes
Calculating -------------------------------------
                     compare-ruby  built-ruby
 hash_aref_dsym_long     117.580M    104.984M bytes -       1.000 times

Comparison:
              hash_aref_dsym_long
          built-ruby: 104984000.0 bytes
        compare-ruby: 117580000.0 bytes - 1.12x  larger
```

Other allocation sites of note

```
==26579== -------------------- 98 of 300 --------------------
==26579== max-live:    323,456 in 2,527 blocks
==26579== tot-alloc:   402,816 in 3,147 blocks (avg size 128.00)
==26579== deaths:      3,147, at avg age 377,614,197 (2.71% of prog lifetime)
==26579== acc-ratios:  0.21 rd, 0.16 wr  (87,620 b-read, 64,622 b-written)
==26579==    at 0x4C2DECF: malloc (in /usr/lib/valgrind/vgpreload_exp-dhat-amd64-linux.so)
==26579==    by 0x1521D3: objspace_xmalloc0 (gc.c:9407)
==26579==    by 0x284A9B: rb_str_buf_new (string.c:1331)
==26579==    by 0x294584: rb_str_inspect (string.c:5911)
==26579==    by 0x2F33F8: vm_call0_cfunc_with_frame (vm_eval.c:86)
==26579==    by 0x2F33F8: vm_call0_cfunc (vm_eval.c:100)
==26579==    by 0x2F33F8: vm_call0_body.constprop.410 (vm_eval.c:132)
==26579==    by 0x2FE4CE: rb_vm_call0 (vm_eval.c:60)
==26579==    by 0x2FE4CE: rb_call0 (vm_eval.c:309)
==26579==    by 0x2FE4CE: rb_call (vm_eval.c:603)
==26579==    by 0x2FE4CE: rb_funcall_with_block (vm_eval.c:857)
==26579==    by 0x2EEFBE: vm_yield_with_symbol (vm_insnhelper.c:2869)
==26579==    by 0x2EEFBE: invoke_block_from_c_bh (vm.c:1131)
==26579==    by 0x2EEFBE: vm_yield (vm.c:1167)
==26579==    by 0x2EEFBE: rb_yield_0 (vm_eval.c:980)
==26579==    by 0x2EEFBE: rb_yield_1 (vm_eval.c:986)
==26579==    by 0x2EEFBE: rb_yield (vm_eval.c:996)
==26579==    by 0x317627: rb_ary_collect_bang (array.c:3052)
==26579==    by 0x2E5B4E: vm_call_cfunc_with_frame (vm_insnhelper.c:2207)
==26579==    by 0x2E5B4E: vm_call_cfunc (vm_insnhelper.c:2225)
==26579==    by 0x2F7D2B: vm_sendish (vm_insnhelper.c:3623)
==26579==    by 0x2F7D2B: vm_exec_core (insns.def:771)
==26579==    by 0x2EE34D: rb_vm_exec (vm.c:1892)
==26579==    by 0x2EEEED: invoke_iseq_block_from_c (vm.c:1104)
==26579==    by 0x2EEEED: invoke_block_from_c_bh (vm.c:1122)
==26579==    by 0x2EEEED: vm_yield (vm.c:1167)
==26579==    by 0x2EEEED: rb_yield_0 (vm_eval.c:980)
==26579==    by 0x2EEEED: rb_yield_1 (vm_eval.c:986)
==26579==    by 0x2EEEED: rb_yield (vm_eval.c:996)
==26579==
==26579== Aggregated access counts by offset:
==26579==
==26579== [   0]  13063 14338 12631 14007 13323 12720 11984 11344 9232 7280 6288 5776 4256 3856 35
84 2976
==26579== [  16]  1968 1216 1296 720 160 48 48 48 64 16 0 0 0 0 0 0
==26579== [  32]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
==26579== [  48]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [  64]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [  80]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [  96]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [ 112]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

==26579== -------------------- 186 of 300 --------------------
==26579== max-live:    155,136 in 1,212 blocks
==26579== tot-alloc:   155,136 in 1,212 blocks (avg size 128.00)
==26579== deaths:      651, at avg age 245,150,551 (1.75% of prog lifetime)
==26579== acc-ratios:  1.63 rd, 0.33 wr  (253,660 b-read, 51,700 b-written)
==26579==    at 0x4C2DECF: malloc (in /usr/lib/valgrind/vgpreload_exp-dhat-amd64-linux.so)
==26579==    by 0x1521D3: objspace_xmalloc0 (gc.c:9407)
==26579==    by 0x284A9B: rb_str_buf_new (string.c:1331)
==26579==    by 0x13AA2E: rb_file_join (file.c:4732)
==26579==    by 0x2E5B4E: vm_call_cfunc_with_frame (vm_insnhelper.c:2207)
==26579==    by 0x2E5B4E: vm_call_cfunc (vm_insnhelper.c:2225)
==26579==    by 0x2F7C7C: vm_sendish (vm_insnhelper.c:3623)
==26579==    by 0x2F7C7C: vm_exec_core (insns.def:789)
==26579==    by 0x2EE34D: rb_vm_exec (vm.c:1892)
==26579==    by 0x2EEEED: invoke_iseq_block_from_c (vm.c:1104)
==26579==    by 0x2EEEED: invoke_block_from_c_bh (vm.c:1122)
==26579==    by 0x2EEEED: vm_yield (vm.c:1167)
==26579==    by 0x2EEEED: rb_yield_0 (vm_eval.c:980)
==26579==    by 0x2EEEED: rb_yield_1 (vm_eval.c:986)
==26579==    by 0x2EEEED: rb_yield (vm_eval.c:996)
==26579==    by 0x374412: dir_yield (dir.c:803)
==26579==    by 0x374412: dir_each_entry (dir.c:860)
==26579==    by 0x374412: dir_each (dir.c:830)
==26579==    by 0x1355D2: rb_ensure (eval.c:1076)
==26579==    by 0x373447: dir_foreach (dir.c:2955)
==26579==    by 0x2E5B4E: vm_call_cfunc_with_frame (vm_insnhelper.c:2207)
==26579==    by 0x2E5B4E: vm_call_cfunc (vm_insnhelper.c:2225)
==26579==
==26579== Aggregated access counts by offset:
==26579==
==26579== [   0]  17075 17890 16143 17188 15855 18234 17616 19067 13136 11474 10379 11350 10094 94
20 8556 7614
==26579== [  16]  5325 5125 5030 4865 3582 2982 3010 3018 2997 3011 3056 3104 2695 2748 2696 2680
==26579== [  32]  1741 1726 1664 1612 1285 1191 1138 1067 1043 1008 984 983 971 985 972 970
==26579== [  48]  565 562 568 559 560 557 557 557 557 557 557 557 557 557 557 557
==26579== [  64]  4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
==26579== [  80]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [  96]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==26579== [ 112]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

==28716== -------------------- 64 of 300 --------------------
==28716== max-live:    273,280 in 2,135 blocks
==28716== tot-alloc:   446,464 in 3,488 blocks (avg size 128.00)
==28716== deaths:      2,277, at avg age 365,758,007 (3.44% of prog lifetime)
==28716== acc-ratios:  0.21 rd, 0.15 wr  (96,380 b-read, 71,208 b-written)
==28716==    at 0x4C2DECF: malloc (in /usr/lib/valgrind/vgpreload_exp-dhat-amd64-linux.so)
==28716==    by 0x1521D3: objspace_xmalloc0 (gc.c:9407)
==28716==    by 0x284A9B: rb_str_buf_new (string.c:1331)
==28716==    by 0x294584: rb_str_inspect (string.c:5911)
==28716==
==28716== Aggregated access counts by offset:
==28716==
==28716== [   0]  14382 15824 13909 15391 14767 14067 13241 12472 10180 7932 6842 6270 4714 4213 3
909 3280
==28716== [  16]  2173 1349 1386 810 216 63 54 55 68 20 1 0 0 0 0 0
==28716== [  32]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [  48]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [  64]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [  80]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [  96]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [ 112]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
==28716== -------------------- 276 of 300 --------------------
==28716== max-live:    19,712 in 154 blocks
==28716== tot-alloc:   146,432 in 1,144 blocks (avg size 128.00)
==28716== deaths:      1,115, at avg age 150,688,929 (1.41% of prog lifetime)
==28716== acc-ratios:  0.06 rd, 0.08 wr  (9,024 b-read, 12,049 b-written)
==28716==    at 0x4C2DECF: malloc (in /usr/lib/valgrind/vgpreload_exp-dhat-amd64-linux.so)
==28716==    by 0x1521D3: objspace_xmalloc0 (gc.c:9407)
==28716==    by 0x284A9B: rb_str_buf_new (string.c:1331)
==28716==    by 0x23B70C: rb_reg_regsub (re.c:3820)
==28716==
==28716== Aggregated access counts by offset:
==28716==
==28716== [   0]  3569 4271 3304 1590 670 640 673 692 537 526 525 518 502 482 465 443
==28716== [  16]  223 203 182 153 127 113 101 85 71 63 57 52 49 48 44 38
==28716== [  32]  14 9 7 6 5 4 4 4 3 1 0 0 0 0 0 0
==28716== [  48]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [  64]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [  80]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [  96]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==28716== [ 112]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

==29686== -------------------- 391 of 600 --------------------
==29686== max-live:    8,192 in 64 blocks
==29686== tot-alloc:   9,472 in 74 blocks (avg size 128.00)
==29686== deaths:      74, at avg age 207,459,973 (1.96% of prog lifetime)
==29686== acc-ratios:  0.21 rd, 0.17 wr  (1,998 b-read, 1,628 b-written)
==29686==    at 0x4C2DECF: malloc (in /usr/lib/valgrind/vgpreload_exp-dhat-amd64-linux.so)
==29686==    by 0x1521D3: objspace_xmalloc0 (gc.c:9407)
==29686==    by 0x284A9B: rb_str_buf_new (string.c:1331)
==29686==    by 0x30BB8A: inspect_ary (array.c:2379)
==29686==
==29686== Aggregated access counts by offset:
==29686==
==29686== [   0]  296 296 296 370 370 370 370 370 296 222 296 74 0 0 0 0
==29686== [  16]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==29686== [  32]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==29686== [  48]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==29686== [  64]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==29686== [  80]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==29686== [  96]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
==29686== [ 112]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Rails specific - redmine boot

Booting redmine only, no real work done otherwise - about 101720 bytes different in current malloc sizes. I understand GC.malloc_allocated_size to reflect the current delta between xmalloc and xfree, but may be wrong. And that value is not representative of total malloc heap churn, judging by the much higher total allocated values coming back from valgrind.

```
lourens@CarbonX1:~/src/redmine$ bundle exec rails c -e production
/home/lourens/src/redmine/vendor/bundle/ruby/2.7.0/gems/activerecord-5.2.1.1/lib/active_record/ass
ociations/builder/collection_association.rb:26: warning: Capturing the given block using Proc.new
is deprecated; use `&block` instead
Loading production environment (Rails 5.2.1.1)
irb(main):001:0> RUBY_DESCRIPTION
=> "ruby 2.7.0dev (2019-04-26 trunk 5689c46457) [x86_64-linux]"
irb(main):002:0> GC.start
=> nil
irb(main):003:0> GC.malloc_allocated_size
=> 74907128

lourens@CarbonX1:~/src/redmine$ bundle exec rails c -e production
/home/lourens/src/redmine/vendor/bundle/ruby/2.7.0/gems/activerecord-5.2.1.1/lib/active_record/ass
ociations/builder/collection_association.rb:26: warning: Capturing the given block using Proc.new
is deprecated; use `&block` instead
Loading production environment (Rails 5.2.1.1)
irb(main):001:0> RUBY_DESCRIPTION
=> "ruby 2.7.0dev (2019-04-26 lower-str-buf-.. 9abd605533) [x86_64-linux]"
```

```
irb(main):002:0> GC.start
=> nil
irb(main):003:0> GC.malloc_allocated_size
=> 74805408
```

Rails specific - some requests / workload

Same sequence of redmine requests - 6 for this branch and trunk respectively using this as a simple initializer for reporting on exit, a 228600 bytes difference.

```
if GC.respond_to?(:malloc_allocated_size)
  at_exit do
    p "#{RUBY_DESCRIPTION} GC.malloc_allocated_size: #{GC.malloc_allocated_size}"
  end
end
```

```
[2019-04-25 23:42:43] INFO  WEBrick 1.4.2
[2019-04-25 23:42:43] INFO  ruby 2.7.0 (2019-04-26) [x86_64-linux]
[2019-04-25 23:42:43] INFO  WEBrick::HTTPServer#start: pid=31335 port=3000
127.0.0.1 - - [25/Apr/2019:23:42:48 WEST] "GET / HTTP/1.1" 200 4373
http://localhost:3000/news -> /
127.0.0.1 - - [25/Apr/2019:23:42:49 WEST] "GET /projects HTTP/1.1" 200 5470
http://localhost:3000/ -> /projects
127.0.0.1 - - [25/Apr/2019:23:42:51 WEST] "GET /activity HTTP/1.1" 200 7373
http://localhost:3000/projects -> /activity
127.0.0.1 - - [25/Apr/2019:23:42:51 WEST] "GET /issues HTTP/1.1" 200 19195
http://localhost:3000/activity -> /issues
127.0.0.1 - - [25/Apr/2019:23:42:52 WEST] "GET /time_entries HTTP/1.1" 200 12508
http://localhost:3000/issues -> /time_entries
127.0.0.1 - - [25/Apr/2019:23:42:53 WEST] "GET /issues/gantt HTTP/1.1" 200 22597
http://localhost:3000/time_entries -> /issues/gantt
127.0.0.1 - - [25/Apr/2019:23:42:54 WEST] "GET /issues/calendar HTTP/1.1" 200 16712
http://localhost:3000/issues/gantt -> /issues/calendar
127.0.0.1 - - [25/Apr/2019:23:42:54 WEST] "GET /news HTTP/1.1" 200 5472
http://localhost:3000/issues/calendar -> /news
^C[2019-04-25 23:42:58] INFO  going to shutdown ...
[2019-04-25 23:42:59] INFO  WEBrick::HTTPServer#start done.
Exiting
"ruby 2.7.0dev (2019-04-26 trunk 5689c46457) [x86_64-linux] GC.malloc_allocated_size: 84901440"
```

```
[2019-04-25 23:41:51] INFO  WEBrick 1.4.2
[2019-04-25 23:41:51] INFO  ruby 2.7.0 (2019-04-26) [x86_64-linux]
[2019-04-25 23:41:51] INFO  WEBrick::HTTPServer#start: pid=31029 port=3000
127.0.0.1 - - [25/Apr/2019:23:41:59 WEST] "GET / HTTP/1.1" 200 4373
http://localhost:3000/activity -> /
127.0.0.1 - - [25/Apr/2019:23:42:02 WEST] "GET /projects HTTP/1.1" 200 5470
http://localhost:3000/ -> /projects
127.0.0.1 - - [25/Apr/2019:23:42:04 WEST] "GET /activity HTTP/1.1" 200 7373
http://localhost:3000/projects -> /activity
127.0.0.1 - - [25/Apr/2019:23:42:05 WEST] "GET /issues HTTP/1.1" 200 19195
http://localhost:3000/activity -> /issues
127.0.0.1 - - [25/Apr/2019:23:42:06 WEST] "GET /time_entries HTTP/1.1" 200 12508
http://localhost:3000/issues -> /time_entries
127.0.0.1 - - [25/Apr/2019:23:42:07 WEST] "GET /issues/gantt HTTP/1.1" 200 22597
http://localhost:3000/time_entries -> /issues/gantt
127.0.0.1 - - [25/Apr/2019:23:42:07 WEST] "GET /javascripts/raphael.js?1543965852 HTTP/1.1" 200 90
648
http://localhost:3000/issues/gantt -> /javascripts/raphael.js?1543965852
127.0.0.1 - - [25/Apr/2019:23:42:08 WEST] "GET /issues/calendar HTTP/1.1" 200 16712
http://localhost:3000/issues/gantt -> /issues/calendar
127.0.0.1 - - [25/Apr/2019:23:42:09 WEST] "GET /news HTTP/1.1" 200 5472
http://localhost:3000/issues/calendar -> /news
^C[2019-04-25 23:42:14] INFO  going to shutdown ...
[2019-04-25 23:42:15] INFO  WEBrick::HTTPServer#start done.
Exiting
"ruby 2.7.0dev (2019-04-26 lower-str-buf-.. 9abd605533) [x86_64-linux] GC.malloc_allocated_size: 8
4672840"
```

History

#1 - 05/22/2019 06:02 AM - ko1 (Koichi Sasada)

Could you give me speed benchmark results, for example with discourse (any rails benchmark is okay)?
If it is difficult, I'll check it.

> Same sequence of redmine requests - 6 for this branch and trunk respectively using this as a simple initializer for reporting on exit, a 228600 bytes difference.

I'm not sure 200KB is good or not. maybe percentage will help.

Thanks,
Koichi

#2 - 07/11/2019 09:24 AM - methodmissing (Lourens Naudé)

Apologies for the late reply - I will do some more testing and qualify with a percentage instead.

#3 - 07/30/2019 04:04 AM - ko1 (Koichi Sasada)

- Assignee set to ko1 (Koichi Sasada)

:)