Ruby master - Feature #12403

Optimise Regexp#match?

05/20/2016 12:43 AM - sam.saffron (Sam Saffron)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

At the moment #match? is dynamically dispatched and args are walked.

Instead

- Add 2 insns instructions for #match? with 1 param and #match? with 2 params
- Amend parser to route #match?("s") and #match?("s", 1) to insns version

Main reason for #match? is performance, might as well make it as fast as possible.

Related issues:

Related to Ruby master - Feature #12306: Implement String #blank? #present? a...	**Open**

History

#1 - 05/20/2016 09:25 AM - naruse (Yui NARUSE)

- Related to Feature #12306: Implement String #blank? #present? and improve #strip and family to handle unicode added

#2 - 05/20/2016 12:47 PM - nobu (Nobuyoshi Nakada)

https://github.com/ruby/ruby/compare/trunk...nobu:feature/12403-optimize-Regexp%23match_p

---

benchmark results:
Execution time (sec)

name	normal	opt
regexp_match_p_match_empty	0.141	0.111
regexp_match_p_match_long	3.941	3.864
regexp_match_p_match_short	0.138	0.099
regexp_match_p_unmatch_long	3.848	3.837
regexp_match_p_unmatch_short	0.103	0.079

Speedup ratio: compare with the result of `normal' (greater is better)

name	opt
regexp_match_p_match_empty	1.266
regexp_match_p_match_long	1.020
regexp_match_p_match_short	1.388
regexp_match_p_unmatch_long	1.003
regexp_match_p_unmatch_short	1.306

#3 - 05/21/2016 01:40 PM - naruse (Yui NARUSE)

After r55102, rb_scan_args is almost statically resolved.

```
000000000051eae0 <rb_reg_match_m_p>:
 *      $&                          #=> nil
 */

static VALUE
rb_reg_match_m_p(int argc, VALUE *argv, VALUE re)
```

```
{
  51eae0:       41 57                   push   %r15
  51eae2:       41 56                   push   %r14
  51eae4:       41 55                   push   %r13
  51eae6:       41 54                   push   %r12
  51eae8:       55                      push   %rbp
  51eae9:       89 fd                   mov    %edi,%ebp
  51eaeb:       53                      push   %rbx
  51eaec:       48 81 ec 88 00 00 00    sub    $0x88,%rsp
  51eaf3:       48 8b 05 86 00 3b 00    mov    0x3b0086(%rip),%rax        # 8ceb80 <_DYNAMIC+0x240>
    VALUE str, initpos;
    long pos = 0;
    regex_t *reg;
    onig_errmsg_buffer err = "";
  51eafa:       48 c7 44 24 10 00 00    movq   $0x0,0x10(%rsp)
  51eb01:       00 00
  51eb03:       48 8d 7c 24 18          lea    0x18(%rsp),%rdi
 *     $&                      #=> nil
 */

static VALUE
rb_reg_match_m_p(int argc, VALUE *argv, VALUE re)
{
  51eb08:       48 8b 08                mov    (%rax),%rcx
  51eb0b:       48 89 4c 24 78          mov    %rcx,0x78(%rsp)
  51eb10:       31 c9                   xor    %ecx,%ecx
    VALUE str, initpos;
    long pos = 0;
    regex_t *reg;
    onig_errmsg_buffer err = "";
  51eb12:       b9 0a 00 00 00          mov    $0xa,%ecx
  51eb17:       31 c0                   xor    %eax,%eax
  51eb19:       f3 48 ab                rep stos %rax,%es:(%rdi)
  51eb1c:       31 c9                   xor    %ecx,%ecx
  51eb1e:       66 89 0f                mov    %cx,(%rdi)
    if (*p != '\0') {
        rb_fatal("bad scan arg format: %s", fmt);
    }
    n_mand = n_lead + n_trail;

    if (argc < n_mand)
  51eb21:       85 ed                   test   %ebp,%ebp
  51eb23:       0f 8e 71 02 00 00       jle    51ed9a <rb_reg_match_m_p+0x2ba>
  51eb29:       48 89 d3                mov    %rdx,%rbx
        }
    }
    /* capture leading mandatory arguments */
    for (i = n_lead; i-- > 0; ) {
        var = vars[vari++];
        if (var) *var = argv[argi];
  51eb2c:       48 8b 3e                mov    (%rsi),%rdi
        argi++;
    }
    /* capture optional arguments */
    for (i = n_opt; i-- > 0; ) {
        var = vars[vari++];
        if (argi < argc - n_trail) {
  51eb2f:       83 fd 01                cmp    $0x1,%ebp
  51eb32:       0f 84 00 01 00 00       je     51ec38 <rb_reg_match_m_p+0x158>
            if (var) *var = argv[argi];
  51eb38:       4c 8b 6e 08             mov    0x8(%rsi),%r13
        else {
            *var = Qnil;
        }
    }

    if (argi < argc) {
  51eb3c:       83 fd 02                cmp    $0x2,%ebp
  51eb3f:       0f 85 55 02 00 00       jne    51ed9a <rb_reg_match_m_p+0x2ba>
    OnigPosition result;
    const UChar *start, *end;
    int tmpreg;

    rb_scan_args(argc, argv, "11", &str, &initpos);
    if (NIL_P(str)) return Qfalse;
```

```
 51eb45:       48 83 ff 08             cmp    $0x8,%rdi
 51eb49:       0f 84 f9 00 00 00       je     51ec48 <rb_reg_match_m_p+0x168>
   str = SYMBOL_P(str) ? rb_sym2str(str) : rb_str_to_str(str);
 51eb4f:       40 80 ff 0c             cmp    $0xc,%dil
 51eb53:       0f 84 cf 01 00 00       je     51ed28 <rb_reg_match_m_p+0x248>
 51eb59:       40 f6 c7 07             test   $0x7,%dil
 51eb5d:       75 17                   jne    51eb76 <rb_reg_match_m_p+0x96>
 51eb5f:       48 f7 c7 f7 ff ff ff    test   $0xfffffffffffffff7,%rdi
 51eb66:       74 0e                   je     51eb76 <rb_reg_match_m_p+0x96>
 51eb68:       8b 07                   mov    (%rdi),%eax
 51eb6a:       83 e0 1f                and    $0x1f,%eax
 51eb6d:       83 f8 14                cmp    $0x14,%eax
 51eb70:       0f 84 b2 01 00 00       je     51ed28 <rb_reg_match_m_p+0x248>
 51eb76:       e8 05 72 04 00          callq  565d80 <rb_str_to_str>
 51eb7b:       49 89 c4                mov    %rax,%r12
```

#4 - 05/22/2016 09:50 PM - sam.saffron (Sam Saffron)

Naruse, I see this was reverted? any way to get rb_scan_args inlined it would be beneficial everywhere.

Nobu, keep in mind your test is looking at 100% hit rate on global method cache, if match? is evicted due to global method cache being open benefit will be greater.

I think this is a great change and hope you can merge in (also =~ is in insns afaik anyway so this add parity)

#5 - 05/24/2016 05:54 AM - naruse (Yui NARUSE)

Sam Saffron wrote:

> Naruse, I see this was reverted? any way to get rb_scan_args inlined it would be beneficial everywhere.

Reverted and reapplied at r55110.