Ruby master - Bug #18511
ruby in infinite loop consuming 100% CPU
01/24/2022 02:13 PM - shenningz (Henning Sackewitz)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>ruby -v:</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN</td>
</tr>
</tbody>
</table>

Description
After running Fluentd (Td-agent) workload for several hours or days, Ruby hangs in an infinite loop.

```
gdb stack trace:
#0 lookup_method_table (id=3377, klass=<optimized out>) at vm_method.c:188
#1 search_method0 (skip_refined=<optimized out>, defined_class_ptr=<optimized out>, id=<optimized out>, klass=<optimized out>) at vm_method.c:747
#2 search_method (defined_class_ptr=<optimized out>, id=<optimized out>, klass=<optimized out>) at vm_method.c:823
#3 method_entry_get_without_cache (defined_class_ptr=<optimized out>, id=<optimized out>, klass=<optimized out>) at vm_method.c:776
#4 method_entry_get (klass=13967557693280, id=entry=3377, defined_class_ptr=defined_class_ptr@entry=0x7f08cbb76210) at vm_method.c:843
#5 0x00007f08d03a00f4 in rb_callable_method_entry (klass=<optimized out>, id=entry=3377) at vm_method.c:891
#6 0x00007f08d03b5942 in rb_search_method_entry (mid=3377, recv=<optimized out>) at vm_eval.c:654
#7 rb_call0 (self=<optimized out>, call_scope=CALL_FCALL, argv=0x7f08cbb76280, argc=1, mid=3377, recv=<optimized out>, ec=0x55cc930e85b0) at vm_eval.c:384
#8 rb_call (scope=CALL_FCALL, argv=0x7f08cbb76280, argc=1, mid=3377, recv=13967557693296) at vm_eval.c:718
#9 rb_funcallv (recv=13967557693296, mid=3377, argc=argc@entry=1, argv=argv@entry=0x7f08cbb76280) at vm_eval.c:958
#10 0x00007f08cc74ff3b in yajl_encode_part (wrapper=wrapper@entry=0x7f08c02ced20, obj=94337115946160, io=io@entry=8) at ../../../../ext/yajl/yajl_ext.c:181
#11 0x00007f08cc750258 in yajl_encode_part (wrapper=wrapper@entry=0x7f08c02ced20, obj=objc@entry=13967547402240, io=io@entry=8) at ../../../../ext/yajl/yajl_ext.c:197
#12 0x00007f08cc750338 in rb_yajl_encoder_encode (argc=argc@entry=1, argv=argv@entry=0x7f08cbb76398, self=<optimized out>) at ../../../../ext/yajl/yajl_ext.c:1115
#13 0x00007f08cc750f8a in rb_yajl_json_ext_hash_to_json (argc=0, argv=0x7f08cbb76fe8, self=<optimized out>) at ../../../../ext/yajl/yajl_ext.c:1179
#14 0x00007f08d039ce8 in vm_call_cfunc_with_frame (empty_kw_splat=<optimized out>, cd=0x55cc92d5f950, calling=<optimized out>, ref_cfp=0x7f08cbb76a40, ec=0x55cc930e85b0) at vm_insnhelper.c:2514
#15 vm_call_cfunc (ec=0x55cc930e85b0, ref_cfp=0x7f08cbb76a40, calling=<optimized out>, cd=0x55cc92d5f950) at vm_insnhelper.c:2539
#16 0x00007f08d039807e in vm_sendish (ec=ec@entry=0x55cc930e85b0, ref_cfp=ref_cfp@entry=0x7f08cbb76a40, cd=0x55cc92d5f950, block_handler=block_handler@entry=0, method_explorer=method_explorer@entry=0x7f08d03a0350 <vm_search_method_wrap>) at vm_insnhelper.c:4023
#17 0x00007f08d03a8c0a in vm_exec_core (ec=<optimized out>) at insns.def:801
#18 0x00007f08d03af000 in rb_vm_exec (ec=0x55cc930e85b0, mjit_enable_p=1) at vm.c:1929
#19 0x00007f08d03bbfa1 in invoke_block (captured=<optimized out>, opt_pc=<optimized out>, type=<optimized out>, cref=<optimized out>, self=<optimized out>, iseq=<optimized out>, ec=<optimized out>) at vm.c:1044
#20 invoke_iseq_block_from_c (me=<optimized out>, is_lambda=<optimized out>, cref=<optimized out>, passed_block_handler=<optimized out>, kw_splat=<optimized out>, arg=<optimized out>, argc=<optimized out>, self=<optimized out>, captured=<optimized out>, ec=<optimized out>) at vm.c:1116
#21 invoke_block_from_c_bh {force_blockarg=<optimized out>, is_lambda=<optimized out>, cref=<optimized out>, passed_block_handler=<optimized out>,
It looks like Ruby is looping here: https://github.com/ruby/ruby/blob/v2_7_5/vm_method.c#L745

A core file (and matching Ruby binaries with debug symbols) can be provided (size 55MB)

History

#1 - 01/24/2022 04:06 PM - jeremyevans0 (Jeremy Evans)
- Status changed from Open to Feedback

This happens inside the yajl-ruby C extension. Can you reproduce this without yajl-ruby?

#2 - 01/24/2022 07:57 PM - shenningz (Henning Sackewitz)

Right now we cannot reproduce this systematically, it just occurs with our Fluentd workload sooner or later. Fluentd Td-agent is packaged this way, we cannot just remove parts from it.

Is there any way to replace the yajl-ruby C extension with another json extension? We already had issues with yajl-ruby before: https://github.com/fluent/fluentd/issues/3547
Is there any way to replace the yajl-ruby C extension with another json extension?

Depending on your needs, maybe the pure Ruby [json_pure](https://rubygems.org/gems/json_pure) gem is good enough. If you have performance requirements, the [oj](https://rubygems.org/gems/oj) gem seems fairly popular and I've heard success stories. It's supposedly faster than yajl-ruby at least for some workloads. Oj uses a C extension, though.

Thank you for the pointers. However, since we are just users and not maintainers of the Fluentd packages, we are not in the position of exchanging components.

To get back to the original issue: could someone look into the core dump? Is there any possibility to upload a 55MB file?

I can't guarantee that anyone is willing to spend time on investigating the core dump, but if you are willing to make publicly available with no strings attached the dump and the relevant binaries (the ruby executable and share libraries), it'll enable such investigations. You can push the dumps to GitHub and post a link here, for example.

If you are looking to make private arrangement with someone so they would keep the information they see in the dump secret, I think that is going to be challenging. Speaking for myself, I wouldn't be willing to engage in such arrangement, especially when unpaid.