[PATCH] Combine call info and cache to speed up method invocation

10/17/2019 01:46 AM - alanwu (Alan Wu)

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

Description
Proposed change: https://github.com/ruby/ruby/pull/2564

To perform a regular method call, the VM needs two structs, rb_call_info and rb_call_cache. At the moment, we allocate these two structures in separate buffers. In the worst case, the CPU needs to read 4 cache lines to complete a method call. Putting the two structures together reduces the maximum number of cache line reads to 2.

Combining the structures also saves 8 bytes per call site as the current layout uses separate pointers for the call info and the call cache. This change saves about 2 MiB on Discourse.

The Optcarrot benchmark receives a performance improvement from this patch. I collected the following results using make install binaries compiled with -DRUBY_NDEBUG, with a sample size of 50 for each category:

<table>
<thead>
<tr>
<th></th>
<th>master-a5245c</th>
<th>after patch</th>
<th>speed-up ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain</td>
<td>42.39</td>
<td>50.17</td>
<td>18.35%</td>
</tr>
<tr>
<td>jit</td>
<td>71.72</td>
<td>72.73</td>
<td>1.41%</td>
</tr>
</tbody>
</table>

These are medium FPS from the benchmark output. For raw benchmark results and basic stats, see https://gist.github.com/XrXr/ce5cb7cf2c3c4d29e58c919fa5c86b33. I took these results with a i7-8750H CPU @ 2.20GHz on a 2018 MacBook Pro. I also ran the benchmark with a AMD 2400G running Arch Linux and observed a 3% improvement without the jit.

Complications

- A new instruction attribute comptime_sp_inc is introduced to calculate SP increase at compile time without using call caches. At compile time, a TS_CALLDATA operand points to a call info struct, but at runtime, the same operand points to a call data struct. Instruction that explicitly define sp_inc also need to define comptime_sp_inc.
- MJIT code for copying call cache becomes slightly more complicated.
- This changes the bytecode format, which might break existing tools.

I think this patch offers a good general performance boost for a manageable amount of code change.

Associated revisions

Revision 89e79976 - 10/24/2019 09:03 AM - alanwu (Alan Wu)
Combine call info and cache to speed up method invocation

To perform a regular method call, the VM needs two structs, rb_call_info and rb_call_cache. At the moment, we allocate these two structures in separate buffers. In the worst case, the CPU needs to read 4 cache lines to complete a method call. Putting the two structures together reduces the maximum number of cache line reads to 2.

Combining the structures also saves 8 bytes per call site as the current layout uses separate two pointers for the call info and the call cache. This saves about 2 MiB on Discourse.
This change improves the Optcarrot benchmark at least 3%. For more details, see attached bugs.ruby-lang.org ticket.

Complications:

- A new instruction attribute `comptime_sp_inc` is introduced to calculate SP increase at compile time without using call caches. At compile time, a `TS_CALLDATA` operand points to a call info struct, but at runtime, the same operand points to a call data struct. Instruction that explicitly define `sp_inc` also need to define `comptime_sp_inc`.
- MJIT code for copying call cache becomes slightly more complicated.
- This changes the bytecode format, which might break existing tools.

[Misc #16258]

History

#1 - 10/17/2019 08:44 AM - ko1 (Koichi Sasada)
- Assignee set to ko1 (Koichi Sasada)

Thank you for your patch.

Conclusion: OK.

Points:

- Current implementation separates ci and cc because of CoW friendliness (ci is immutable data and cc is mutable data). However, there are no measurements how it affect on CoW friendliness. Because ci is immutable data, we can pre-compile these data and it will improve startup time. However, there are no implementation of it.
- For Guild, I will rewrite inline cache (cc) because of atomicity. However, Ruby 2.7 doesn't have this change. For Ruby 2.7 only this patch is accepted.

#2 - 10/24/2019 09:04 AM - alanwu (Alan Wu)
- Status changed from Open to Closed

Applied in changeset git|89e7997622038f82115f34dbb4ea382e02bed163.

Combine call info and cache to speed up method invocation

To perform a regular method call, the VM needs two structs, `rb_call_info` and `rb_call_cache`. At the moment, we allocate these two structures in separate buffers. In the worst case, the CPU needs to read 4 cache lines to complete a method call. Putting the two structures together reduces the maximum number of cache line reads to 2.

Combining the structures also saves 8 bytes per call site as the current layout uses separate two pointers for the call info and the call cache. This saves about 2 MiB on Discourse.

This change improves the Optcarrot benchmark at least 3%. For more details, see attached bugs.ruby-lang.org ticket.

Complications:

- A new instruction attribute `comptime_sp_inc` is introduced to calculate SP increase at compile time without using call caches. At compile time, a `TS_CALLDATA` operand points to a call info struct, but at runtime, the same operand points to a call data struct. Instruction that explicitly define `sp_inc` also need to define `comptime_sp_inc`.
- MJIT code for copying call cache becomes slightly more complicated.
- This changes the bytecode format, which might break existing tools.

[Misc #16258]