Ruby master - Feature #14954
Add :wait option to RubyVM::MJIT.pause

07/31/2018 10:10 AM - k0kubun (Takashi Kokubun)

Status: Closed  Priority: Normal
Assignee: Target version:

Description
Problem
To measure an MJIT-generated code's performance, currently we need to do sleep ...; RubyVM::MJIT.pause and it's hard to decide an appropriate sleep seconds.

Solution
Add an :wait option (default: true) and:

```ruby
# Wait for finishing all of queued compilations, and pause MJIT worker (different from the current behavior).
# This will be convenient for most of micro benchmarks. So I wanna make it default. It will be breaking but MJIT.pause is not released anywhere.
RubyVM::MJIT.pause
RubyVM::MJIT.pause(wait: true)

# Wait only for a currently-compiled method, and pause MJIT worker immediately (the current behavior).
# This will be convenient when there are many methods, like on Rails.
RubyVM::MJIT.pause(wait: false)
```

Related issues:
Related to Ruby master - Feature #14830: RubyVM::MJIT.pause / RubyVM::MJIT.resume added

History
#1 - 07/31/2018 10:12 AM - k0kubun (Takashi Kokubun)
- Related to Feature #14830: RubyVM::MJIT.pause / RubyVM::MJIT.resume added

#2 - 08/09/2018 09:58 AM - k0kubun (Takashi Kokubun)
- Status changed from Open to Closed

committed in r64250

#3 - 08/09/2018 10:12 AM - Eregon (Benoit Daloze)
Just a thought:
Pausing MJIT for benchmarks, even microbenchmarks, might have quite unintended effects once there is some profiling (e.g. branch profiling, type profiling, value profiling, etc) or if the JIT is sensitive to the calling context (for example with inlining).
With any of those, the actual measurement phase might look different than the warmup phase to the JIT and as a result run the benchmarked method(s) with the non-JIT version of the method(s).
Ideally the warmup phase looks identical to the measurement phase but this is quite tricky to achieve and there certainly are differences in e.g. benchmark-driver and benchmark-ips currently.

#4 - 08/09/2018 11:59 AM - k0kubun (Takashi Kokubun)

```
Once there is some profiling (e.g. branch profiling, type profiling, value profiling, etc)
```

This part didn't make sense, probably because such profiling is not introduced yet. Could you provide some example?

```
or if the JIT is sensitive to the calling context (for example with inlining).
```
I can understand what you mean, but I believe it doesn't matter if we don't replace calling context by OSR.

Even with those issues, what MJIT.pause is solving can't be solved without MJIT.pause, and we would just need to make effort to prepare an appropriate warmup script.

With any of those, the actual measurement phase might look different than the warmup phase to the JIT and as a result run the benchmarked method(s) with the non-JIT version of the method(s).

To measure MJIT's performance correctly, the measured entity needs to be the same method(s) anyway. So the entity should be already JIT-ed on the measurement phase regardless of whether MJIT worker is paused or not.

Ideally the warmup phase looks identical to the measurement phase but this is quite tricky to achieve and there certainly are differences in e.g. benchmark-driver and benchmark-ips currently.

In benchmark-driver whose measured entity is a top-level script (never JIT-ed anyway for now), this doesn't matter for the above reasons. For benchmark-ips whose measured entity is a block, I believe the block is identical in warmup and measurement phase, and thus the same block is already JIT-ed on warmup and pausing it doesn't prevent the block on measurement phase from being JIT-ed, if my assumption about benchmark-ips is correct.