

Ruby - Feature #14104

Remove `trace` instructions

11/14/2017 08:40 AM - ko1 (Koichi Sasada)

Status:	Closed	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	
Target version:	2.5	

Description

Abstract

Remove trace instructions from bytecode and it will improve performance, 10% faster.
Instead of trace instruction, we add event information for each instruction.
In other words, we unified trace instructions with the following instructions.

Backgroud

TracePoint and old set_trace_func method need to know which point they need to kick hooks at.
trace VM instruction is inserted to represent such hook points and kicks hooks if needed.
It is easy to introduce hook points into bytecode.
However, this technique introduces some overhead on "no-hooks" situations (and I believe 99% of workloads does not need hooks, especially on production environment).

For example, compiled code of the following method has several trace instructions.

```
def foo x
  _y = x
end
```

Disasm output is here:

```
== disasm: #<ISeq:foo@<compiled>>=====
local table (size: 2, argc: 1 [opts: 0, rest: -1, post: 0, block: -1, kw: -1@-1, kwrest: -1])
[ 2] x<Arg>      [ 1] _y
0000 trace      8 ( 1)
0002 trace      1
0004 getlocal   x, 0
0007 dup
0008 setlocal   _y, 0
0011 trace     16 ( 2)
0013 leave      ( 1)
```

```
# disasm code is:
puts RubyVM::InstructionSequence.compile("def foo x
  _y = x
end", nil, nil, 0, trace_instruction: true, operands_unification: false).disasm
```

You can see 3 trace instructions on 0000, 0002 and 0011. They means hook points for :call, :line and :return events. It enables to support TracePoint, but introduces 3 dispatches for useless instructions.

We can remove trace instructions with the trace_instruction: true compile option, but we can't turn on TracePoint for such compiled ISeqs.

Proposal

Remove trace instruction and introduce event flags as information for each instructions.

With same code above, we can get the following disasm code.

```
== disasm: #<ISeq:foo@<compiled>>=====
local table (size: 2, argc: 1 [opts: 0, rest: -1, post: 0, block: -1, kw: -1@-1, kwrest: -1])
```

```
[ 2] x<Arg>      [ 1] _y
0000 getlocal    x, 0                      ( 1) [LiCa]
0003 dup
0004 setlocal    _y, 0
0007 leave       [Re]
```

0000 shows [LiCa], it means :line and :call events.

0007 shows [Re] which represents a :return event.

It is too heavy if we check these event flags on each instruction dispatch.

So we introduce new instructions: trace_prefix instructions such as trace_getlocal and so on.

All normal instructions have corresponding trace_prefix instructions (generated by the VM generator automatically).

If we need to enable TracePoint (or set_trace_func), then rewrite all of instruction sequences (ISeqs) in the heap from normal instructions to trace prefix instructions dynamically. trace_prefix instructions check flags for an executing instruction and kick hooks if needed.

- Good:
 - a. Low overhead for normal execution because we don't need to dispatch useless trace instructions.
 - b. We don't have trace instructions, so that it is easy to turn on tail call optimization flags.
 - c. We can enable/disable traces on small units.
- Bad:
 - a. Big overhead to turn on/off TracePoint because we need to modify all of existing ISeqs.
 - b. Introducing a few incompatibilities for tracing.

For (Bad-a) I believe nobody turn on and turn off tracepoints many times, so that turning on/off overhead is not so big impact.

For (Good-c), I need to explain more.

Current TracePoint enables all of hooks for all methods, ISeqs. With this technique, however, we can select enable/disable the hooks in more small units, like files, classes, methods or a line. For example, we can enable the hooks only method named foo. Now we have no interface to specify it, but we can design later.

Evaluation

```
require 'benchmark'
```

```
def foo n
end
```

```
N = 100_000_000
```

```
Benchmark.bm(10) { |x|
  x.report('trace off') {
    N.times {
      foo(10)
      foo(10)
      foo(10)
    }
  }
  x.report('trace on') {
    TracePoint.new{}.enable
    N.times {
      foo(10)
      foo(10)
      foo(10)
    }
  }
}
```

Current trunk:

	user	system	total	real
trace off	9.090678	0.000000	9.090678	(9.083007)
trace on	108.217320	0.005524	108.222844	(108.201941)

modified:

	user	system	total	real
trace off	6.647247	0.000000	6.647247	(6.641530)

```
trace on      93.405389    0.000000   93.405389  ( 93.404864)
```

Compatibility issues

With this change, we introduced a few incompatibilities:

- Line numbers on `:return/:b_return` events show the last executed lines, instead of end lines (without return statement).
- begin block doesn't show `:line` events.

Related issues:

Related to Ruby - Bug #14582: Unable to use ``method__entry`` and ``method_retur...`

[Open](#)

Associated revisions

Revision 665ba24b446971fdf652a9c57c32b176d0018636 - 11/14/2017 12:58 PM - ko1 (Koichi Sasada)

remove trace instruction. [Feature #14104]

- `tool/instruction.rb`: create `trace_` prefix instructions.
- `compile.c` (`ADD_TRACE`): do not add trace instructions but add `TRACE` link elements. `TRACE` elements will be unified with a next instruction as instruction information.
- `vm_trace.c` (`update_global_event_hook`): modify all `ISeqs` when hooks are enabled.
- `iseq.c` (`rb_iseq_trace_set`): added to toggle `trace_` instructions.
- `vm_insnhelper.c` (`vm_trace`): added.
This function is a body of `trace_` prefix instructions.
- `vm_insnhelper.h` (`JUMP`): save PC to a control frame.
- `insns.def` (`trace`): removed.
- `vm_exec.h` (`INSN_ENTRY_SIG`): add debug output (disabled).

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60763 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 665ba24b - 11/14/2017 12:58 PM - ko1 (Koichi Sasada)

remove trace instruction. [Feature #14104]

- `tool/instruction.rb`: create `trace_` prefix instructions.
- `compile.c` (`ADD_TRACE`): do not add trace instructions but add `TRACE` link elements. `TRACE` elements will be unified with a next instruction as instruction information.
- `vm_trace.c` (`update_global_event_hook`): modify all `ISeqs` when hooks are enabled.
- `iseq.c` (`rb_iseq_trace_set`): added to toggle `trace_` instructions.
- `vm_insnhelper.c` (`vm_trace`): added.
This function is a body of `trace_` prefix instructions.
- `vm_insnhelper.h` (`JUMP`): save PC to a control frame.
- `insns.def` (`trace`): removed.
- `vm_exec.h` (`INSN_ENTRY_SIG`): add debug output (disabled).

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60763 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision c08e8886badd47890a54bdc54f1c09de7ad5c8e8 - 12/20/2017 04:24 AM - mame (Yusuke Endoh)

`compile.c`: add a `RUBY_EVENT_COVERAGE_LINE` event for line coverage

2.5's line coverage measurement was about two times slower than 2.4 because of two reasons; (1) `vm_trace` uses `rb_iseq_event_flags` (which

takes $O(n)$ currently where n is the length of `iseq`) to get an event type, and (2) `RUBY_EVENT_LINE` uses `setjmp` to call an event hook.

This change adds a special event for line coverage, `RUBY_EVENT_COVERAGE_LINE`, and adds `tracecoverage` instructions where the event occurs in `iseq`.

`tracecoverage` instruction calls an event hook without `vm_trace`.

And, `RUBY_EVENT_COVERAGE_LINE` is an internal event which does not use `setjmp`.

This change also cancels `lineno` change due to the deletion of trace instructions [Feature #14104]. So fixes [Bug #14191].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61350 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision c08e8886 - 12/20/2017 04:24 AM - mame (Yusuke Endoh)

compile.c: add a `RUBY_EVENT_COVERAGE_LINE` event for line coverage

2.5's line coverage measurement was about two times slower than 2.4 because of two reasons; (1) `vm_trace` uses `rb_iseq_event_flags` (which takes $O(n)$ currently where n is the length of `iseq`) to get an event type, and (2) `RUBY_EVENT_LINE` uses `setjmp` to call an event hook.

This change adds a special event for line coverage, `RUBY_EVENT_COVERAGE_LINE`, and adds `tracecoverage` instructions where the event occurs in `iseq`.

`tracecoverage` instruction calls an event hook without `vm_trace`.

And, `RUBY_EVENT_COVERAGE_LINE` is an internal event which does not use `setjmp`.

This change also cancels `lineno` change due to the deletion of trace instructions [Feature #14104]. So fixes [Bug #14191].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61350 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 919fa894f5e0fed105ad1513cb8be0748ae46295 - 12/23/2017 11:00 AM - ko1 (Koichi Sasada)

revert line number spec of `*return` events.

- `compile.c` (`rb_iseq_compile_node`): line number spec of `:return`, `:b_return` and `:end` events of 'TracePoint' is changed for [Feature #14104].

Quoted from [Feature #14104]:

Line numbers on `:return`/`:b_return` events show the last executed lines, instead of end lines (without return statement).
Note that `:end` event also affected.

However, "buybug", a well-known ruby debugger depends on previous behavior so that I reverted this specification.

- `test/ruby/test_settracefunc.rb`: catch up this fix.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61424 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 919fa894 - 12/23/2017 11:00 AM - ko1 (Koichi Sasada)

revert line number spec of `*return` events.

- `compile.c` (`rb_iseq_compile_node`): line number spec of `:return`, `:b_return` and `:end` events of 'TracePoint' is changed for [Feature #14104].

Quoted from [Feature #14104]:

Line numbers on `:return`/`:b_return` events show the last executed lines, instead of end lines (without return statement).
Note that `:end` event also affected.

However, "buybug", a well-known ruby debugger depends on previous behavior so that I reverted this specification.

- test/ruby/test_settracefunc.rb: catch up this fix.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61424 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision bd80fc53159f4ada3549b6f85209af13be375095 - 06/05/2018 09:19 PM - Eric Wong

test/dtrace: enable Tracepoint for function calls

Since [Feature #14104], "trace" instructions are no longer emitted by default, so we must enable them explicitly for function tracing to work.

[ruby-core:85965]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63580 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision bd80fc53159f4ada3549b6f85209af13be375095 - 06/05/2018 09:19 PM - Eric Wong

test/dtrace: enable Tracepoint for function calls

Since [Feature #14104], "trace" instructions are no longer emitted by default, so we must enable them explicitly for function tracing to work.

[ruby-core:85965]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63580 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision bd80fc53 - 06/05/2018 09:19 PM - Eric Wong

test/dtrace: enable Tracepoint for function calls

Since [Feature #14104], "trace" instructions are no longer emitted by default, so we must enable them explicitly for function tracing to work.

[ruby-core:85965]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63580 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 100bf2757468439106775a7d95a791a8c10b874a - 08/22/2018 11:09 AM - mame (Yusuke Endoh)

compile.c: remove tracecoverage instruction for line coverage

Line coverage was based on special instruction "tracecoverage". Now, instead, it uses the mechanism of trace hook [Feature #14104].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64509 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 100bf2757468439106775a7d95a791a8c10b874a - 08/22/2018 11:09 AM - mame (Yusuke Endoh)

compile.c: remove tracecoverage instruction for line coverage

Line coverage was based on special instruction "tracecoverage". Now, instead, it uses the mechanism of trace hook [Feature #14104].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64509 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 100bf275 - 08/22/2018 11:09 AM - mame (Yusuke Endoh)

compile.c: remove tracecoverage instruction for line coverage

Line coverage was based on special instruction "tracecoverage". Now, instead, it uses the mechanism of trace hook [Feature #14104].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64509 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 6c9a7050329e1da977319e55aa9728c9b50f8250 - 10/20/2018 10:45 AM - mame (Yusuke Endoh)

Remove tracecoverage instructions

The instructions were used only for branch coverage. Instead, it now uses a trace framework [Feature #14104].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@65225 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 6c9a7050329e1da977319e55aa9728c9b50f8250 - 10/20/2018 10:45 AM - mame (Yusuke Endoh)

Remove tracecoverage instructions

The instructions were used only for branch coverage.
Instead, it now uses a trace framework [Feature #14104].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@65225 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 6c9a7050 - 10/20/2018 10:45 AM - mame (Yusuke Endoh)

Remove tracecoverage instructions

The instructions were used only for branch coverage.
Instead, it now uses a trace framework [Feature #14104].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@65225 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 11/14/2017 09:08 AM - shevegen (Robert A. Heiler)

	user	system	total	real
trace off	9.090678	0.000000	9.090678 (	9.083007)
trace on	108.217320	0.005524	108.222844	(108.201941)

	user	system	total	real
trace off	6.647247	0.000000	6.647247 (	6.641530)
trace on	93.405389	0.000000	93.405389	(93.404864)

\o/

#2 - 11/14/2017 10:11 AM - ko1 (Koichi Sasada)

- Description updated

#3 - 11/14/2017 11:44 AM - ko1 (Koichi Sasada)

Here is a result of benchmark result with

```
ruby ~/ruby/src/trunk/benchmark/driver.rb -e ruby_2_4::~~/ruby/install/ruby_2_4/bin/ruby -e trunk_oct::~~/ruby/i
ninstall/trunk_r60079/bin/ruby -e trunk::~~/ruby/install/trunk/bin/ruby -e modified::~~/ruby/install/gitruby/bin/r
uby -v -r 5
```

<https://gist.github.com/anonymous/64b7e73e5972befc33a9acde09612f90>

#4 - 11/14/2017 12:58 PM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

Applied in changeset trunk|r60763.

remove trace instruction. [Feature #14104]

- tool/instruction.rb: create trace_ prefix instructions.
- compile.c (ADD_TRACE): do not add trace instructions but add TRACE link elements. TRACE elements will be unified with a next instruction as instruction information.
- vm_trace.c (update_global_event_hook): modify all ISeqs when hooks are enabled.
- iseq.c (rb_iseq_trace_set): added to toggle trace_ instructions.
- vm_inshelper.c (vm_trace): added.
This function is a body of trace_ prefix instructions.
- vm_inshelper.h (JUMP): save PC to a control frame.
- insns.def (trace): removed.

- `vm_exec.h` (`INSN_ENTRY_SIG`): add debug output (disabled).

#5 - 11/14/2017 04:40 PM - vmakarov (Vladimir Makarov)

ko1 (Koichi Sasada) wrote:

Abstract

Remove trace instructions from bytecode and it will improve performance, 10% faster.
Instead of trace instruction, we add event information for each instruction.
In other words, we unified trace instructions with the following instructions.

Hi, Koichi. I am very glad that you addressed this issue. The current implementation of tracing is just wasting CPU time. I thought about this too but my approach would be a different because I thought that I had to maintain the full compatibility of trace behaviour.

So I considered to remove and insert trace insns when the tracing is off or on. It requires to change branch offset, catch table offsets, and line tables. To speed up this, I would keep the log of pairs (`<place in iseq code, catch table, or line table>`, `)` and log of pairs (`,` `)`.

I also thought that such general log mechanism would be useful in possible speculative iseq optimizations on MRI insns level.

But if the full trace behaviour compatibility is not so important, your solution is much much simpler and practically has the same code locality as you keep codes of trace and usual insns in disjoint parts of the interpreter switch-stmt.

I think rebuilding iseq is not a problem. Sorry if I missed the code change but I don't see a check of version of loaded iseq on compatibility. So what will happen if somebody tries to load the old iseq with trace insns into a new version of MRI?

#6 - 11/27/2017 05:59 AM - ko1 (Koichi Sasada)

vmakarov (Vladimir Makarov) wrote:

So I considered to remove and insert trace insns when the tracing is off or on. It requires to change branch offset, catch table offsets, and line tables. To speed up this, I would keep the log of pairs (`<place in iseq code, catch table, or line table>`, `)` and log of pairs (`,` `)`.

I also thought that such general log mechanism would be useful in possible speculative iseq optimizations on MRI insns level.

I completely agree your approach is more compatible and flexible for other optimization techniques. But it is some tough work to implement it because PC info are located variety of places. I also believe (sometime) we need to implement similar technique you explained.

But if the full trace behaviour compatibility is not so important, your solution is much much simpler and practically has the same code locality as you keep codes of trace and usual insns in disjoint parts of the interpreter switch-stmt.

Yes, simplicity is the reason why I use this technique.
Only few days I needed to make.

I think rebuilding iseq is not a problem. Sorry if I missed the code change but I don't see a check of version of loaded iseq on compatibility. So what will happen if somebody tries to load the old iseq with trace insns into a new version of MRI?

We don't guarantee the cross-version dump/load compatibility (this is why we embed version info into dumped results). So I believe there are no problem on this change.

#7 - 06/09/2022 05:36 AM - mame (Yusuke Endoh)

- Related to Bug #14582: Unable to use ``method__entry`` and ``method_return`` tracing probes since 2.5 added