

How does YJIT work in --enable-shared case?

11/10/2022 03:50 PM - jaruga (Jun Aruga)

Status:	Closed	
Priority:	Normal	
Assignee:		
Description We are trying to add the new YJIT feature that is ported to Rust[1][2] in Ruby 3.2 RPM on Fedora project.[3] I am trying to understand how the YJIT works in the case the ./configure --enable-yjit --enable-shared. In the case below, when building Ruby on the latest master branch 199b59f065ce6f1c13b8424f35a70c513523211b, the static libraries libruby-static.a and ./libruby-static.a were built. <pre><mock-chroot> sh-5.2\$ cat /etc/fedora-release Fedora release 38 (Rawhide) <mock-chroot> sh-5.2\$./autogen.sh <mock-chroot> sh-5.2\$./configure --prefix=\$HOME/local/ruby-yjit-199b59f065 --enable-shared --enable-yjit 2>&1 tee configure.log ... * MJIT support: yes * YJIT support: yes <mock-chroot> sh-5.2\$ make <mock-chroot> sh-5.2\$ find . -name "*.a" ./yjit/target/release/libyjit.a ./libruby-static.a After running make install, these static libraries (*.a files) were not copied to the installed directory. That makes sense, as Ruby works with shared libraries (*.so files). <mock-chroot> sh-5.2\$ make install 2>&1 tee make_install.log <mock-chroot> sh-5.2\$ find ~/local/ruby-yjit-199b59f065/ -name "*.a" => empty In this case, we really don't need the libyjit.a to run the YJIT right? I couldn't find the .so file something like yjit.so. Which .so file contains the YJIT feature (maybe the content of the yjit/src/**/*.rs)? <mock-chroot> sh-5.2\$ find ~/local/ruby-yjit-199b59f065/ -name "*.so" grep yjit.so => empty How can we test the content of the yjit/src/**/*.rs? For example, if the command below works, I can say that I tested the content of the yjit/src/**/*.rs? <mock-chroot> sh-5.2\$ which rustc /usr/bin/rustc <mock-chroot> sh-5.2\$ rustc --version rustc 1.65.0 (Fedora 1.65.0-1.fc38) <mock-chroot> sh-5.2\$ ~/local/ruby-yjit-199b59f065/bin/ruby --yjit -e 'puts "abc"' abc</pre> References • [1] https://bugs.ruby-lang.org/issues/18481 • [2] https://github.com/ruby/ruby/commit/f90549cd38518231a6a74432fe1168c943a7cc18#diff-1d53751ddf3ffeb25cfe609c6bd28746651e23ea81d51fe582b0f635f0896e0d • [3] https://src.fedoraproject.org/rpms/ruby/pull-request/139		

History

#1 - 11/10/2022 04:49 PM - jaruga (Jun Aruga)

- Description updated

#2 - 11/10/2022 05:03 PM - jaruga (Jun Aruga)

How can we test the content of the `yjit/src/**/*.rs`? For example, if the command below works, I can say that I tested the content of the `yjit/src/**/*.rs`?

Here is a bit old commit on the master branch 131c31a9209c61f84d318aa18b61f468f48b8219. It was compiled with `./configure --enable-yjit=dev --enable-shared --prefix=$HOME/local/ruby-yjit-dev-so-131c31a920`. Can I say that I executed the content of the `yjit/src/**/*.rs`?

```
$ which ruby
~/local/ruby-yjit-dev-so-131c31a920/bin/ruby

$ ruby --yjit --yjit-call-threshold=1 --yjit-stats -e 'puts "abc"'
abc
***YJIT: Printing YJIT statistics on exit***
method call exit reasons:
  (all relevant counters are zero)
invokesuper exit reasons:
  (all relevant counters are zero)
leave exit reasons:
  interp_return      1 (100.0%)
getblockparamproxy exit reasons:
  (all relevant counters are zero)
getinstancevariable exit reasons:
  (all relevant counters are zero)
setinstancevariable exit reasons:
  (all relevant counters are zero)
opt_aref exit reasons:
  (all relevant counters are zero)
expandarray exit reasons:
  (all relevant counters are zero)
opt_getinlinecache exit reasons:
  (all relevant counters are zero)
invalidation reasons:
  (all relevant counters are zero)
bindings_allocations:      73
bindings_set:              0
compiled_block_count:      8
compiled_iseq_count:       4
compiled_page_count:       1
freed_iseq_count:          0
freed_page_count:          0
invalidation_count:        0
constant_state_bumps:      0
inline_code_size:          1047
outlined_code_size:        643
freed_code_size:           0
code_gc_count:             0
num_gc_obj_refs:           9
total_exit_count:          1
total_insns_count:         233888
vm_insns_count:            233876
yjit_insns_count:          12
ratio_in_yjit:             0.0%
avg_len_in_yjit:           12.0
total_exits:               0
```

#3 - 11/10/2022 05:18 PM - jaruga (Jun Aruga)

Sorry for my repeated posting. I uninstalled the `rustc` to check if the Rust compiler is really used in the YJIT process. And the commands worked without the `rust` command. Could you tell me why it worked?

```
<mock-chroot> sh-5.2$ cat /etc/fedora-release
Fedora release 38 (Rawhide)
```

```
<mock-chroot> sh-5.2# dnf remove rustc
```

```
<mock-chroot> sh-5.2$ which rustc
which: no rustc in (/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/sbin)

<mock-chroot> sh-5.2$ ~/local/ruby-yjit-199b59f065/bin/ruby --yjit --yjit-call-threshold=1 -e '100.times { |n|
puts "Hello #{n}" }'
...
Hello 98
Hello 99

<mock-chroot> sh-5.2$ ~/local/ruby-yjit-199b59f065/bin/ruby --yjit --yjit-call-threshold=1 -e '10000.times { |
n| puts "Hello #{n}" }'
...
Hello 9998
Hello 9999
```

#4 - 11/10/2022 07:01 PM - ufuk (Ufuk Kayserilioglu)

jaruga (Jun Aruga) wrote in [#note-3](#):

... the commands worked without the rust command. Could you tell me why it worked?

I can answer this part of the question.

```
<mock-chroot> sh-5.2$ which rustc
which: no rustc in (/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/sbin)

<mock-chroot> sh-5.2$ ~/local/ruby-yjit-199b59f065/bin/ruby --yjit --yjit-call-threshold=1 -e '100.times {
|n| puts "Hello #{n}" }'
...
Hello 98
Hello 99
```

YJIT does NOT generate Rust code at runtime to perform its JIT compilation. YJIT is, itself, a Rust code-base that generates the correct assembly code for optimized methods at runtime. So rustc is only needed to compile the YJIT code itself, once you've compiled Ruby with YJIT, there is no dependency on rustc at runtime anymore.

#5 - 11/10/2022 09:10 PM - k0kubun (Takashi Kokubun)

- Status changed from Open to Closed

In this case, we really don't need the libyjit.a to run the YJIT right?

Try running make with V=1. You can confirm libyjit.a is linked in the build process (as SOLIBS :p) even with --enable-shared. I don't think YJIT has ever done anything special for --enable-shared; it always builds and uses libyjit.a. libyjit.a doesn't seem to depend on any Rust-related runtime, so removing the rustc package wouldn't impact its behavior. As [@ufuk \(Ufuk Kayserilioglu\)](#) said, it doesn't rely on rustc(1) at runtime either.

#6 - 11/11/2022 02:11 PM - jaruga (Jun Aruga)

Guys, thanks for answering and sharing the info!

YJIT does NOT generate Rust code at runtime to perform its JIT compilation. YJIT is, itself, a Rust code-base that generates the correct assembly code for optimized methods at runtime. So rustc is only needed to compile the YJIT code itself, once you've compiled Ruby with YJIT, there is no dependency on rustc at runtime anymore.

Ah, okay. I misundertood the YJIT behavior. I was thinking that the rustc was executed in the JIT process, that is like the gcc or clang is executed in the MJIT process ([#17817](#)).

Try running make with V=1. You can confirm libyjit.a is linked in the build process (as SOLIBS :p) even with --enable-shared. I don't think YJIT has ever done anything special for --enable-shared; it always builds and uses libyjit.a. libyjit.a doesn't seem to depend on any Rust-related runtime, so removing the rustc package wouldn't impact its behavior. As [@ufuk \(Ufuk Kayserilioglu\)](#) (Ufuk Kayserilioglu) said, it doesn't rely on rustc(1) at runtime either.

Thanks for giving the info!

I checked it with the latest master branch 90bbc891b192c30432c517ccb279ed687bb2d0b4 now. And I was able to see the libyjit.a is a part of the SOLIBS in the Makefile, an that the libyjit.a was actually linked to create the libruby.so.3.2.0, miniruby and ruby binary files.

```
$ ./autogen.sh
```

```

$ ./configure --prefix=$HOME/local/ruby-yjit-90bbc891b1 --enable-shared --enable-yjit 2>&1 | tee configure.log
$ make V=1 2>&1 | tee make_v1.log

$ vi Makefile
...
YJIT_LIBS=yjit/target/release/libyjit.a
...
SOLIBS = $(MAINLIBS)
...
MAINLIBS = $(YJIT_LIBS) -lz -lrt -lrt -lgmp -ldl -lcrypt -lm -lpthread
...

$ grep libyjit.a make_v1.log
SOLIBS = yjit/target/release/libyjit.a -lz -lrt -lrt -lgmp -ldl -lcrypt -lm -lpthread
touch yjit/target/release/libyjit.a
gcc -O3 -fno-fast-math -ggdb3 -Wall -Wextra -Wdeprecated-declarations -Wdiv-by-zero -Wduplicated-cond -Wimplicit-function-declaration -Wimplicit-int -Wmisleading-indentation -Wpointer-arith -Wwrite-strings -Wold-style-definition -Wimplicit-fallthrough=0 -Wmissing-noreturn -Wno-cast-function-type -Wno-constant-logical-operand -Wno-long-long -Wno-missing-field-initializers -Wno-overlength-strings -Wno-packed-bitfield-compat -Wno-parentheses-equality -Wno-self-assign -Wno-tautological-compare -Wno-unused-parameter -Wno-unused-value -Wsuggest-attribute=format -Wsuggest-attribute=noreturn -Wunused-variable -Wundef -fPIC -L. -fstack-protector-strong -rdynamic -Wl,-export-dynamic -Wl,--no-as-needed -fstack-protector-strong main.o dmydln.o miniinit.o dmyext.o array.o ast.o bignum.o class.o compar.o compile.o complex.o cont.o debug.o debug_counter.o dir.o dln_find.o encoding.o enum.o enumerator.o error.o eval.o file.o gc.o hash.o inits.o io.o io_buffer.o iseq.o load.o marshal.o math.o memory_view.o mjit.o mjit_compiler.o node.o numeric.o object.o pack.o parse.o proc.o process.o ractor.o random.o range.o rational.o re.o regcomp.o regenc.o regerror.o regex.o regparse.o regsyntax.o ruby.o scheduler.o shape.o signal.o sprintf.o st.o strftime.o string.o struct.o symbol.o thread.o time.o transcode.o transient_heap.o util.o variable.o version.o vm.o vm_backtrace.o vm_dump.o vm_sync.o vm_trace.o yjit.o coroutine/amd64/Context.o probes.o enc/ascii.o enc/us_ascii.o enc/unicode.o enc/utf_8.o enc/trans/newline.o setproctitle.o strlcat.o strlcpy.o addr2line.o yjit/target/release/libyjit.a -lz -lrt -lrt -lgmp -ldl -lcrypt -lm -lpthread -lm -lpthread -o miniruby
: 'merging yjit/target/release/libyjit.a into libruby-static.a' && \
(cd libyjit/ && gcc-ar -x ../yjit/target/release/libyjit.a) && \
+ : 'merging yjit/target/release/libyjit.a into libruby-static.a'
+ gcc-ar -x ../yjit/target/release/libyjit.a
gcc -shared -Wl,--compress-debug-sections=zlib -Wl,-soname,libruby.so.3.2 -fstack-protector-strong dln.o locale_init.o loadpath.o array.o ast.o bignum.o class.o compar.o compile.o complex.o cont.o debug.o debug_counter.o dir.o dln_find.o encoding.o enum.o enumerator.o error.o eval.o file.o gc.o hash.o inits.o io.o io_buffer.o iseq.o load.o marshal.o math.o memory_view.o mjit.o mjit_compiler.o node.o numeric.o object.o pack.o parse.o proc.o process.o ractor.o random.o range.o rational.o re.o regcomp.o regenc.o regerror.o regex.o regparse.o regsyntax.o ruby.o scheduler.o shape.o signal.o sprintf.o st.o strftime.o string.o struct.o symbol.o thread.o time.o transcode.o transient_heap.o util.o variable.o version.o vm.o vm_backtrace.o vm_dump.o vm_sync.o vm_trace.o yjit.o coroutine/amd64/Context.o probes.o enc/ascii.o enc/us_ascii.o enc/unicode.o enc/utf_8.o enc/trans/newline.o setproctitle.o strlcat.o strlcpy.o addr2line.o builtin.o dmyext.o dmyenc.o yjit/target/release/libyjit.a -lz -lrt -lrt -lgmp -ldl -lcrypt -lm -lpthread -o libruby.so.3.2.0
gcc -O3 -fno-fast-math -ggdb3 -Wall -Wextra -Wdeprecated-declarations -Wdiv-by-zero -Wduplicated-cond -Wimplicit-function-declaration -Wimplicit-int -Wmisleading-indentation -Wpointer-arith -Wwrite-strings -Wold-style-definition -Wimplicit-fallthrough=0 -Wmissing-noreturn -Wno-cast-function-type -Wno-constant-logical-operand -Wno-long-long -Wno-missing-field-initializers -Wno-overlength-strings -Wno-packed-bitfield-compat -Wno-parentheses-equality -Wno-self-assign -Wno-tautological-compare -Wno-unused-parameter -Wno-unused-value -Wsuggest-attribute=format -Wsuggest-attribute=noreturn -Wunused-variable -Wundef -fPIC -L. -fstack-protector-strong -rdynamic -Wl,-export-dynamic -Wl,--no-as-needed -fstack-protector-strong main.o -Wl,-rpath,/build/libruby-yjit-90bbc891b1/lib -L/build/libruby-yjit-90bbc891b1/lib -lruby yjit/target/release/libyjit.a -lz -lrt -lrt -lgmp -ldl -lcrypt -lm -lpthread -lm -lpthread -o ruby

```