Ruby trunk - Feature #11788

New ISeq serialize binary format

12/08/2015 10:56 AM - ko1 (Koichi Sasada)

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

Description

Abstract

I wrote a new RubyVM::InstructionSequence (ISeq) object serializer and de-serializer binary format.
Matz had approved to introduce this feature to Ruby 2.3 as *experimental* feature.
So I'll commit them.

There are two methods to serialize and de-serialize.

- RubyVM::InstructionSequence#to_binary_format returns binary format data as String object.
- RubyVM::InstructionSequence.from_binary_format(data) de-serialize it.

The goal of this project is to provide "machine dependent" binary file to achieve:

- fast bootstrap time for big applications
- reduce memory consumption with several techniques

"Machine dependent" means you can't migrate compiled binaries to other machines.

They are not goals of this project:

- packing scripts to one package
- migrate obfuscate binary to other node to hide source code

To achieve such goals, we need to consider compatibility issues such as __FILE__, __dir__, DATA, and so on (for example, consider about this code: Dir.glob(File.join(__dir__, '*.rb')).

This proposal doesn't contain "how to store compiled binaries".
For example, Rubinius makes *.rbc file automatically.
However, Matz does not like such automatic compilation.

So that my proposal only show user storage class interface.
People can try to make your own ISeq binary storage.

For example,

- making a compiled binary files automatically in same directory of script files like Rubinius,
- store compiled binaries in some DB
- make storage data structure in your own.

I wrote several samples:

- dbm: use dbm
- fs: [default] use file system. locate compiled file in same directory of script file like Rubinius. foo.rb.yarb will be created for foo.rb.
- fs2: use file system. locate compiled file in specified directory.
- nothing: do nothing.

You can see my sample implementation:
https://github.com/ko1/ruby/blob/iseq_p1/sample/iseq_loader.rb

The key interface is RubyVM::InstructionSequence.load_iseq(fname).
When MRI try to load any script named fname, then call this method with fname if defined.
The return value is an ISeq object, then MRI use this ISeq object instead of parsing/compiling fname file.

Note that this proposal is "experimental".
These interfaces are only for experiments.
For example, if we want to use several binary storage,
this interface doesn't support multiple storage (lack of extensibility).

Current status

The current implementation is not matured because the binary size is very big because pointer size consumes 32/64 bits.
It is easy to reduce, but I remain this weak point.

Now, one goal "reduction of memory consumption" is not achieved because no techniques are introduced to share/unload or something.
This is future work.

Evaluation

Several evaluation results:

resolv.rb

Try to load resolv.rb 1,000 times (and remove Resolv class each time).

```
compile     12.360000   0.310000  12.670000 ( 13.413011)
compile     12.120000   0.300000  12.420000 ( 13.195313)
compile     12.230000   0.270000  12.500000 ( 13.242140)

eager load
load         3.750000   0.180000   3.930000 (  3.918169)
load         4.000000   0.170000   4.170000 (  4.178442)
load         4.120000   0.200000   4.320000 (  4.320233)

lazy load
load         2.410000   0.090000   2.500000 (  2.609716)
load         2.280000   0.210000   2.490000 (  2.518892)
load         2.310000   0.110000   2.420000 (  2.419687)
```

3.25 times faster than normal compilation.
If we use lazy loading technique, it is 5.2 times faster.

fileutils.rb

Try similar to resolv.rb.

```
                 user     system      total        real
compile      8.540000   0.130000   8.670000 (  8.703615)
compile      8.540000   0.150000   8.690000 (  8.693870)
compile      8.430000   0.120000   8.550000 (  8.547480)

eager load
load         4.470000   0.150000   4.620000 (  4.659934)
load         4.500000   0.140000   4.640000 (  4.640365)
load         4.610000   0.100000   4.710000 (  4.708825)

lazy load
load         3.510000   0.140000   3.650000 (  3.694146)
load         3.470000   0.130000   3.600000 (  3.609040)
load         3.550000   0.150000   3.700000 (  3.831015)
```

Only 1.8 times faster (eager) and 2.4 times faster (lazy).
This is because the initialization of FileUtils class run long time.
It uses module_eval(str) to add methods.

Simple rails application

run time rails r '' on simple Rails application (https://github.com/ko1/tracer_demo_rails_app tracers are disabled).

```
compile:
real    0m2.049s
user    0m1.601s
sys     0m0.402s

eager:
real    0m1.544s
user    0m1.094s
sys     0m0.422s

lazy:
$ time rails r ''
real    0m1.536s
user    0m1.112s
sys     0m0.388s
```

Not so impressive result. It seems there are many initialization code.

Associated revisions

Revision 3dbb3901 - 12/08/2015 01:58 PM - ko1 (Koichi Sasada)

- introduce new ISeq binary format serializer/de-serializer
  and a pre-compilation/runtime loader sample.
  [Feature #11788]

- iseq.c: add new methods:
  - RubyVM::InstructionSequence#to_binary_format(extra_data = nil)
  - RubyVM::InstructionSequence.from_binary_format(binary)
  - RubyVM::InstructionSequence.from_binary_format_extra_data(binary)

- compile.c: implement body of this new feature.

- load.c (rb_load_internal0), iseq.c (rb_iseq_load_iseq):
  call RubyVM::InstructionSequence.load_iseq(fname) with
  loading script name if this method is defined.

We can return any ISeq object as a result value.
Otherwise loading will be continue as usual.

This interface is not matured and is not extensible.
So that we don't guarantee the future compatibility of this method.
Basically, you should'nt use this method.

- iseq.h: move ISEQ_MAJOR/MINOR_VERSION (and some definitions)
  from iseq.c.

- encoding.c (rb_data_is_encoding), internal.h: added.

- vm_core.h: add several supports for lazy load.
  - add USE_LAZY_LOAD macro to specify enable or disable of this feature.
  - add several fields to rb_iseq_t.
  - introduce new macro rb_iseq_check().

- insns.def: some check for lazy loading feature.

- vm_insnhelper.c: ditto.

- proc.c: ditto.

- vm.c: ditto.

- test/lib/iseq_loader_checker.rb: enabled iff suitable

environment variables are provided.

- test/runner.rb: enable lib/iseq_loader_checker.rb.
- sample/iseq_loader.rb: add sample compiler and loader.

  $ ruby sample/iseq_loader.rb [dir]

will compile all ruby scripts in [dir].
With default setting, this compile creates *.rb.yarb files
in same directory of target .rb scripts.

```
$ ruby -r sample/iseq_loader.rb [app]
```

will run with enable to load compiled binary data.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@52949 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 52949 - 12/08/2015 01:58 PM - ko1 (Koichi Sasada)

- introduce new ISeq binary format serializer/de-serializer
  and a pre-compilation/runtime loader sample.
  [Feature #11788]
- iseq.c: add new methods:
  - RubyVM::InstructionSequence#to_binary_format(extra_data = nil)
  - RubyVM::InstructionSequence.from_binary_format(binary)
  - RubyVM::InstructionSequence.from_binary_format_extra_data(binary)
- compile.c: implement body of this new feature.
- load.c (rb_load_internal0), iseq.c (rb_iseq_load_iseq):
  call RubyVM::InstructionSequence.load_iseq(fname) with
  loading script name if this method is defined.

We can return any ISeq object as a result value.
Otherwise loading will be continue as usual.

This interface is not matured and is not extensible.
So that we don't guarantee the future compatibility of this method.
Basically, you should'nt use this method.

- iseq.h: move ISEQ_MAJOR/MINOR_VERSION (and some definitions)
  from iseq.c.
- encoding.c (rb_data_is_encoding), internal.h: added.
- vm_core.h: add several supports for lazy load.
  - add USE_LAZY_LOAD macro to specify enable or disable of this feature.
  - add several fields to rb_iseq_t.
  - introduce new macro rb_iseq_check().
- insns.def: some check for lazy loading feature.
- vm_insnhelper.c: ditto.
- proc.c: ditto.
- vm.c: ditto.
- test/lib/iseq_loader_checker.rb: enabled iff suitable
  environment variables are provided.
- test/runner.rb: enable lib/iseq_loader_checker.rb.
- sample/iseq_loader.rb: add sample compiler and loader.

$ ruby sample/iseq_loader.rb [dir]

will compile all ruby scripts in [dir].
With default setting, this compile creates *.rb.yarb files
in same directory of target .rb scripts.

```
$ ruby -r sample/iseq_loader.rb [app]
```

will run with enable to load compiled binary data.

Revision 52949 - 12/08/2015 01:58 PM - ko1 (Koichi Sasada)

- introduce new ISeq binary format serializer/de-serializer
  and a pre-compilation/runtime loader sample.
  [Feature #11788]
- iseq.c: add new methods:
  - RubyVM::InstructionSequence#to_binary_format(extra_data = nil)
  - RubyVM::InstructionSequence.from_binary_format(binary)
  - RubyVM::InstructionSequence.from_binary_format_extra_data(binary)
- compile.c: implement body of this new feature.
- load.c (rb_load_internal0), iseq.c (rb_iseq_load_iseq):
  call RubyVM::InstructionSequence.load_iseq(fname) with
  loading script name if this method is defined.

We can return any ISeq object as a result value.
Otherwise loading will be continue as usual.

This interface is not matured and is not extensible.
So that we don't guarantee the future compatibility of this method.
Basically, you should'nt use this method.

- iseq.h: move ISEQ_MAJOR/MINOR_VERSION (and some definitions)
  from iseq.c.
- encoding.c (rb_data_is_encoding), internal.h: added.
- vm_core.h: add several supports for lazy load.
  - add USE_LAZY_LOAD macro to specify enable or disable of this feature.
  - add several fields to rb_iseq_t.
  - introduce new macro rb_iseq_check().
- insns.def: some check for lazy loading feature.
- vm_insnhelper.c: ditto.
- proc.c: ditto.
- vm.c: ditto.
- test/lib/iseq_loader_checker.rb: enabled iff suitable
  environment variables are provided.
- test/runner.rb: enable lib/iseq_loader_checker.rb.
- sample/iseq_loader.rb: add sample compiler and loader.

  $ ruby sample/iseq_loader.rb [dir]

will compile all ruby scripts in [dir].
With default setting, this compile creates *.rb.yarb files
in same directory of target .rb scripts.

```
$ ruby -r sample/iseq_loader.rb [app]
```

will run with enable to load compiled binary data.

Revision 52949 - 12/08/2015 01:58 PM - ko1 (Koichi Sasada)

- introduce new ISeq binary format serializer/de-serializer
  and a pre-compilation/runtime loader sample.
  [Feature #11788]
- iseq.c: add new methods:
  - RubyVM::InstructionSequence#to_binary_format(extra_data = nil)
  - RubyVM::InstructionSequence.from_binary_format(binary)
  - RubyVM::InstructionSequence.from_binary_format_extra_data(binary)
- compile.c: implement body of this new feature.
- load.c (rb_load_internal0), iseq.c (rb_iseq_load_iseq):
  call RubyVM::InstructionSequence.load_iseq(fname) with
  loading script name if this method is defined.

We can return any ISeq object as a result value.
Otherwise loading will be continue as usual.

This interface is not matured and is not extensible.
So that we don't guarantee the future compatibility of this method.
Basically, you should'nt use this method.

- iseq.h: move ISEQ_MAJOR/MINOR_VERSION (and some definitions)
  from iseq.c.
- encoding.c (rb_data_is_encoding), internal.h: added.
- vm_core.h: add several supports for lazy load.
  - add USE_LAZY_LOAD macro to specify enable or disable of this feature.
  - add several fields to rb_iseq_t.
  - introduce new macro rb_iseq_check().
- insns.def: some check for lazy loading feature.
- vm_insnhelper.c: ditto.
- proc.c: ditto.
- vm.c: ditto.
- test/lib/iseq_loader_checker.rb: enabled iff suitable
  environment variables are provided.
- test/runner.rb: enable lib/iseq_loader_checker.rb.
- sample/iseq_loader.rb: add sample compiler and loader.

  $ ruby sample/iseq_loader.rb [dir]

will compile all ruby scripts in [dir].
With default setting, this compile creates *.rb.yarb files
in same directory of target .rb scripts.


```
$ ruby -r sample/iseq_loader.rb [app]
```

will run with enable to load compiled binary data.

Revision 52949 - 12/08/2015 01:58 PM - ko1 (Koichi Sasada)

- introduce new ISeq binary format serializer/de-serializer
  and a pre-compilation/runtime loader sample.
  [Feature #11788]

- iseq.c: add new methods:

  - RubyVM::InstructionSequence#to_binary_format(extra_data = nil)
  - RubyVM::InstructionSequence.from_binary_format(binary)
  - RubyVM::InstructionSequence.from_binary_format_extra_data(binary)

- compile.c: implement body of this new feature.

- load.c (rb_load_internal0), iseq.c (rb_iseq_load_iseq):
  call RubyVM::InstructionSequence.load_iseq(fname) with
  loading script name if this method is defined.

We can return any ISeq object as a result value.
Otherwise loading will be continue as usual.

This interface is not matured and is not extensible.
So that we don't guarantee the future compatibility of this method.
Basically, you should'nt use this method.

- iseq.h: move ISEQ_MAJOR/MINOR_VERSION (and some definitions)
  from iseq.c.

- encoding.c (rb_data_is_encoding), internal.h: added.

- vm_core.h: add several supports for lazy load.

  - add USE_LAZY_LOAD macro to specify enable or disable of this feature.
  - add several fields to rb_iseq_t.
  - introduce new macro rb_iseq_check().

- insns.def: some check for lazy loading feature.

- vm_insnhelper.c: ditto.

- proc.c: ditto.

- vm.c: ditto.

- test/lib/iseq_loader_checker.rb: enabled iff suitable
  environment variables are provided.

- test/runner.rb: enable lib/iseq_loader_checker.rb.

- sample/iseq_loader.rb: add sample compiler and loader.

  $ ruby sample/iseq_loader.rb [dir]

will compile all ruby scripts in [dir].
With default setting, this compile creates *.rb.yarb files
in same directory of target .rb scripts.

```
$ ruby -r sample/iseq_loader.rb [app]
```

will run with enable to load compiled binary data.

Revision 52949 - 12/08/2015 01:58 PM - ko1 (Koichi Sasada)

- introduce new ISeq binary format serializer/de-serializer
  and a pre-compilation/runtime loader sample.
  [Feature #11788]

- iseq.c: add new methods:

    - RubyVM::InstructionSequence#to_binary_format(extra_data = nil)
    - RubyVM::InstructionSequence.from_binary_format(binary)
    - RubyVM::InstructionSequence.from_binary_format_extra_data(binary)

- compile.c: implement body of this new feature.
- load.c (rb_load_internal0), iseq.c (rb_iseq_load_iseq):
  call RubyVM::InstructionSequence.load_iseq(fname) with
  loading script name if this method is defined.

We can return any ISeq object as a result value.
Otherwise loading will be continue as usual.

This interface is not matured and is not extensible.
So that we don't guarantee the future compatibility of this method.
Basically, you should'nt use this method.

- iseq.h: move ISEQ_MAJOR/MINOR_VERSION (and some definitions)
  from iseq.c.
- encoding.c (rb_data_is_encoding), internal.h: added.
- vm_core.h: add several supports for lazy load.
    - add USE_LAZY_LOAD macro to specify enable or disable of this feature.
    - add several fields to rb_iseq_t.
    - introduce new macro rb_iseq_check().
- insns.def: some check for lazy loading feature.
- vm_insnhelper.c: ditto.
- proc.c: ditto.
- vm.c: ditto.
- test/lib/iseq_loader_checker.rb: enabled iff suitable
  environment variables are provided.
- test/runner.rb: enable lib/iseq_loader_checker.rb.
- sample/iseq_loader.rb: add sample compiler and loader.

  $ ruby sample/iseq_loader.rb [dir]

will compile all ruby scripts in [dir].
With default setting, this compile creates *.rb.yarb files
in same directory of target .rb scripts.

```
$ ruby -r sample/iseq_loader.rb [app]
```

will run with enable to load compiled binary data.

History

#1 - 12/08/2015 11:48 AM - normalperson (Eric Wong)

ko1@atdot.net wrote:

- RubyVM::InstructionSequence#to_binary_format returns binary format data as String object.
- RubyVM::InstructionSequence.from_binary_format(data) de-serialize it.

Why a new custom binary format instead of existing iseq.to_a + marshal?
Is performance improved enough to be worth extra code?

#2 - 12/08/2015 12:55 PM - ko1 (Koichi Sasada)

Good question. I never take notice about that.

Try resolv.rb with use to_a/load_iseq.

```
                 user     system      total        real
compile     10.590000   0.270000  10.860000 ( 11.452102)
compile     10.580000   0.250000  10.830000 ( 11.455050)
compile     10.630000   0.330000  10.960000 ( 11.580943)

Use iseq_load()
load        26.380000   0.690000  27.070000 ( 27.768315)
load        27.220000   0.660000  27.880000 ( 28.576489)
load        29.860000   0.630000  30.490000 ( 31.242912)
```

To use to_a, we need to use Marshal.dump().
And loading also needs Marshal.dump and ISeq.load.

> Is performance improved enough to be worth extra code?

Yes.

#3 - 12/08/2015 01:59 PM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

Applied in changeset r52949.

- introduce new ISeq binary format serializer/de-serializer
  and a pre-compilation/runtime loader sample.
  [Feature #11788]

- iseq.c: add new methods:
  - RubyVM::InstructionSequence#to_binary_format(extra_data = nil)
  - RubyVM::InstructionSequence.from_binary_format(binary)
  - RubyVM::InstructionSequence.from_binary_format_extra_data(binary)

- compile.c: implement body of this new feature.

- load.c (rb_load_internal0), iseq.c (rb_iseq_load_iseq):
  call RubyVM::InstructionSequence.load_iseq(fname) with
  loading script name if this method is defined.

We can return any ISeq object as a result value.
Otherwise loading will be continue as usual.

This interface is not matured and is not extensible.
So that we don't guarantee the future compatibility of this method.
Basically, you should'nt use this method.

- iseq.h: move ISEQ_MAJOR/MINOR_VERSION (and some definitions)
  from iseq.c.

- encoding.c (rb_data_is_encoding), internal.h: added.

- vm_core.h: add several supports for lazy load.
  - add USE_LAZY_LOAD macro to specify enable or disable of this feature.
  - add several fields to rb_iseq_t.
  - introduce new macro rb_iseq_check().

- insns.def: some check for lazy loading feature.

- vm_insnhelper.c: ditto.

- proc.c: ditto.

- vm.c: ditto.

- test/lib/iseq_loader_checker.rb: enabled iff suitable
  environment variables are provided.

- test/runner.rb: enable lib/iseq_loader_checker.rb.

- sample/iseq_loader.rb: add sample compiler and loader.

  $ ruby sample/iseq_loader.rb [dir]

will compile all ruby scripts in [dir].
With default setting, this compile creates *.rb.yarb files
in same directory of target .rb scripts.

```
$ ruby -r sample/iseq_loader.rb [app]
```

will run with enable to load compiled binary data.

#4 - 12/09/2015 12:53 AM - ko1 (Koichi Sasada)

- Status changed from Closed to Assigned

Note that current implementation lacks error checking and verification,
so that broken binary data cause SEGV (or access overflow) easily.

This is another reason why I wrote it is not for "migration" purpose.
(malicious binary data can come from outside)
Same reason why we don't publish rb_iseq_load() as Ruby method.

It will be a security risk if malicious person can pass modified binary data to MRI.

#5 - 12/09/2015 04:41 AM - usa (Usaku NAKAMURA)

Koichi Sasada wrote:

> Note that current implementation lacks error checking and verification,
> so that broken binary data cause SEGV (or access overflow) easily.
> (snip)
> It will be a security risk if malicious person can pass modified binary data to MRI.

You should mention it at NEWS.

#6 - 12/09/2015 05:08 AM - ko1 (Koichi Sasada)

On 2015/12/09 13:41, usa@garbagecollect.jp wrote:

> You should mention it at NEWS.

Updated.
I'm not sure how detail description is needed on an entry.

--
// SASADA Koichi at atdot dot net

#7 - 12/09/2015 05:14 AM - usa (Usaku NAKAMURA)

Koichi Sasada wrote:

> Updated.

Thanks.

> I'm not sure how detail description is needed on an entry.

I'm not sure, too.
But I can say that if a feature introduces a known security risk, users must be informed with documentations.

#8 - 12/11/2015 08:28 AM - ko1 (Koichi Sasada)

On 2015/12/11 17:15, Vít Ondruch wrote:

> Dne 8.12.2015 v 11:56 ko1@atdot.net napsal(a):
>
> > The goal of this project is to provide "machine dependent" binary file to achieve:

> Could you please elaborate more about this? For example on Fedora/RHEL, Python byte code is shipped precompiled per platform, is this machine dependent code more fine grained then Python byte code, so this would not work?

You are right.
This proposal doesn't support "shipped precompiled per platform".

- making a compiled binary files automatically in same directory of script files like Rubinius,

> This does not work for packaged Ruby applications, i.e. they are installed somewhere in read-only /usr, so it should go probably somewhere into /var/cache IMO. Or it should be configurable somehow during runtime with some fallback paths.

Yes.

--
// SASADA Koichi at atdot dot net

#9 - 04/11/2016 06:28 AM - ko1 (Koichi Sasada)

- Status changed from Assigned to Closed

MRI 2.3 was shipped with this feature.