Ruby trunk - Feature #15743

RubyVM should be renamed to CRuby

04/02/2019 11:53 AM - Eregon (Benoit Daloze)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	Next Major

Description

My understanding is that RubyVM is supposed to only exist on MRI/CRuby, and not on any other Ruby implementation.
In fact, almost all features under RubyVM are not realistically implementable on other major Ruby implementations (JRuby, TruffleRuby, Rubinius).

Unfortunately, the name doesn't reflect that it is specific to CRuby, e.g., JRuby/TruffleRuby/Rubinius are also "Ruby VMs".
And as a result it is not clear for many Ruby users that RubyVM is CRuby-specific.

The documentation is also unclear:

```
The RubyVM module provides some access to Ruby internals. This module is
for very limited purposes, such as debugging, prototyping, and research.
 Normal users must not use it.
```

So I propose to rename RubyVM to CRuby.
That way, it is clear for everyone that this module is CRuby-specific.
It's also consistent with the JRuby module, the TruffleRuby module and the Rubinius module.

Proposed migration path:

- Introduce CRuby as an alias of RubyVM on trunk, and deprecate the RubyVM constant on trunk (for 2.7).
- Remove RubyVM in Ruby 3.0.

What do you think?

cc ko1 (Koichi Sasada)k0kubun (Takashi Kokubun)headius (Charles Nutter)

Related issues:

Related to Ruby trunk - Feature #15752: A dedicated module for experimental f...	**Open**

History

#1 - 04/02/2019 11:56 AM - Eregon (Benoit Daloze)

- Description updated

#2 - 04/02/2019 03:08 PM - shevegen (Robert A. Heiler)

I am not involved in any of this so this should be up to those who are involved; but one comment that is somewhat unrelated is that, at the least to me, the meaning of the words are different. RubyVM sort of indicates some VM, whereas CRuby refers to ... what name exactly? C code of Ruby?

The names don't quite seem to convey the same meaning in my opinion.

I don't refer to your other points, which may or may not be fine, but from a conceptual point of view, I think RubyVM is a better name than CRuby. But that's just a side comment - it's up to those who handle the code of course. (You need to keep in mind that existing code such as RubyVM::InstructionSequence.compile() would eventually become e. g. CRuby::InstructionSequence.compile() and it just reads differently to me.)

#3 - 04/02/2019 05:37 PM - headius (Charles Nutter)

I agree with making the "RubyVM" namespace a little less generic. In JRuby and TruffleRuby, we both have our own "JRuby" and "TruffleRuby" namespaces, which tuck away our implementation-specific stuff.

However I'm not sure CRuby is really the right name. As far as I have seen, CRuby has only ever been an informal name, like MRI. Honestly I think

the best option would be to stick with the original VM name and call it YARV. It's presenting YARV instruction sequences, YARV-specific VM metrics and tweakables, YARV-specific heap and GC calls, and so on.

Or some other official name that could be CRuby?

#4 - 04/02/2019 06:39 PM - chrisseaton (Chris Seaton)

> whereas CRuby refers to ... what name exactly?

CRuby is the name some people (including Matz I believe) used to refer to the standard implementation of Ruby. Other people use the name MRI. It literally means 'the Ruby VM implemented in C'. But it's an established name, it isn't just some random thing Benoit made up.

#5 - 04/03/2019 12:36 AM - mame (Yusuke Endoh)

I'm negative against the rename for some reasons.

1) CRuby/MRI is a popular name, not an official name. In fact, RUBY_ENGINE is "ruby".

```
$ ruby -e 'p RUBY_ENGINE'
"ruby"
```

2) A casual user is not recommended to use RubyVM. You, a designer of JRuby, TruffleRuby, etc., can (or should) ignore a library that is using RubyVM.

3) Some APIs in RubyVM, e.g., RubyVM::AbstractSyntaxTree, might be compatible among some implementations, if you want. If the namespace is split, a user must follow them.

4) The name of CRuby is already used: https://rubygems.org/gems/cruby

I think the rename is worse than worth.

Of course, JRuby, TruffleRuby, etc. are all "Ruby VMs". I think you can use the namespace "RubyVM" as you like. A library that is truly serious to use RubyVM will (or should) check RUBY_ENGINE. Or, if you are still worried, you may want to use a namespace RubyVM::JRuby and RubyVM::TruffleRuby.

#6 - 04/03/2019 11:01 AM - k0kubun (Takashi Kokubun)

Could you provide real-world use case in which CRuby looks more appropriate than RubyVM?

Off topic: mruby is also "the Ruby VM implemented in C" and "Matz' Ruby implementation"

#7 - 04/03/2019 12:05 PM - Eregon (Benoit Daloze)

mame (Yusuke Endoh) wrote:

> I'm negative against the rename for some reasons.

Thank you for your feedback. I will reply inline.

> 1) CRuby/MRI is a popular name, not an official name. In fact, RUBY_ENGINE is "ruby".

Yes, ruby, not Ruby. For me Ruby means the language, and ruby is CRuby/MRI's executable.
So RubyVM feels wrong to me, because it does not mean (at least in the current situation), a general-purpose "Ruby language VM namespace".

Maybe we should have such a common namespace for experimenting with portable experimental features, but that is another discussion.
It certainly shouldn't be named RubyVM as that would introduce a lot of confusion.

> 2) A casual user is not recommended to use RubyVM. You, a designer of JRuby, TruffleRuby, etc., can (or should) ignore a library that is using RubyVM.

But we cannot, end users use whatever is available in MRI, including RubyVM. Some gems use RubyVM.
Thinking end users will only use what they should is a dream, the reality is alternative Ruby implementations need to implement as much as possible like CRuby for compatibility.
And it's not always obvious RubyVM makes their code non-portable, so they should only use it as last resort,
knowing it will likely never work that way on other Ruby implementations.

Similarly problematic: TracePoint#instruction_sequence, which is a core method, returns a RubyVM::InstructionSequence, which can only work on CRuby.

> 3) Some APIs in RubyVM, e.g., RubyVM::AbstractSyntaxTree, might be compatible among some implementations, if you want.

Let's see:

- RubyVM::InstructionSequence is a representation of the MRI/CRuby/YARV bytecode. I don't think we can reproduce this on e.g., TruffleRuby, where there is no Ruby bytecode involved.
- RubyVM::AbstractSyntaxTree exposes the internal CRuby AST. I'm pretty sure other Ruby implementations have a slightly different AST while parsing. Ripper seems the portable version of this.
- RubyVM::MJIT is obviously specific to MJIT, not a general JIT (e.g., RubyVM::MJIT.pause doesn't make much sense without MJIT).
- RubyVM::OPTS, DEFAULT_PARAMS and INSTRUCTION_NAMES all return MRI-specific information, which make little sense for other Ruby implementations (e.g., INSTRUCTION_NAMES with no bytecode?).
- RubyVM.stat returns a Hash containing MRI-specific values, such as global_method_state, which simply doesn't exist on TruffleRuby.

This is very clear: basically none of these APIs were designed with Ruby implementation portability in mind.
And this is fine, I am not against MRI experimenting with MRI-specific APIs,
but I would like them to be clearly marked as such, and RubyVM doesn't achieve that.

> If the namespace is split, a user must follow them.

I think that is a good thing, implementation-specific parts would be clear in the code.

> 4) The name of CRuby is already used: https://rubygems.org/gems/cruby

I think we can ignore that, we're not creating a gem, and that seems an old unmaintained rubyforge project.

> Of course, JRuby, TruffleRuby, etc. are all "Ruby VMs". I think you can use the namespace "RubyVM" as you like.

I think that would be confusing, as currently basically everything under RubyVM is MRI-specific (and not implementable on TruffleRuby).
Anyway, I'm fine to define things on the TruffleRuby module for now,
my real concern is it's not clear for users and even for us that RubyVM is CRuby-specific, and I want to fix that.

> A library that is truly serious to use RubyVM will (or should) check RUBY_ENGINE. Or, if you are still worried, you may want to use a namespace RubyVM::JRuby and RubyVM::TruffleRuby.

But I would expect many don't, because they think RubyVM makes sense for "the Ruby language" while it's actually only meaningful on CRuby.

#8 - 04/03/2019 12:09 PM - Eregon (Benoit Daloze)

k0kubun (Takashi Kokubun) wrote:

> Could you provide real-world use case in which CRuby looks more appropriate than RubyVM?

I think all constants under RubyVM are clear examples:

```
CRuby::InstructionSequence - it is the bytecode of CRuby
CRuby::AbstractSyntaxTree  - it is the AST of CRuby
CRuby::MJIT                - it is the method JIT of CRuby
```

or if you like YARV better:

```
YARV::InstructionSequence
YARV::AbstractSyntaxTree
YARV::MJIT
```

All these things are CRuby-specific and not portable, so qualifying them clearly is not only more precise, it's more correct IMHO.

#9 - 04/03/2019 12:16 PM - hsbt (Hiroshi SHIBATA)

I strongly against to rename to CRuby from RubyVM. Because the implementation language like C in the namespace is irrelevant for users of Ruby.

#10 - 04/03/2019 12:18 PM - Eregon (Benoit Daloze)

About the naming discussion, I picked CRuby because I think this is what matz (Yukihiro Matsumoto) uses when referring to this implementation.
matz (Yukihiro Matsumoto) Could you confirm?

MRI is the more well-known name, but it also has other meanings, and some consider it to be the "ruby 1.8 interpreter" and YARV "ruby 1.9+ VM".

I think CRuby is simply more general than MRI and YARV, e.g., it's not clear if MJIT is part of MRI or YARV, but it's certainly part of CRuby.
YARV is notably used as a name for the bytecode, which is an implementation detail of CRuby.

If the bytecode format changes significantly (e.g., imagine something like RTL),
then it's still CRuby::AbstractSyntaxTree and CRuby::InstructionSequence but YARV::InstructionSequence would be confusing.
I also don't think it's needed to rename this module if the bytecode format changes.

#11 - 04/03/2019 12:21 PM - Eregon (Benoit Daloze)

hsbt (Hiroshi SHIBATA) wrote:

> I strongly against to rename to CRuby from RubyVM. Because the implementation language like C in the namespace is irrelevant for users of Ruby.

Would you prefer MRI or YARV then?
As I mentioned above RubyVM seems highly confusing for something MRI-specific, and not a general "Ruby VM" namespace.

#12 - 04/04/2019 01:43 AM - mame (Yusuke Endoh)

Eregon,

Thank you for your reply, but I'm not convinced yet.

> But we cannot, end users use whatever is available in MRI, including RubyVM. Some gems use RubyVM.

If you think so, you have to implement the feature in any way, regardles of whether it is renamed to CRuby or not? I don't see at all what the renaming will solve.

- RubyVM::InstructionSequence is a representation of the MRI/CRuby/YARV bytecode. I don't think we can reproduce this on e.g., TruffleRuby, where there is no Ruby bytecode involved.
- RubyVM::AbstractSyntaxTree exposes the internal CRuby AST. I'm pretty sure other Ruby implementations have a slightly different AST while parsing. Ripper seems the portable version of this.

I believe that they can be implemented as compatibility layers. For example, TruffleRuby's RubyVM::InstructionSequence can return a MRI-compatible bytecode, but TruffleRuby will not use the bytecode for execution. I think that such a dummy API would be actually useful for some cases, e.g., MRI bytecode-level static analyzer.

You may think it is not worthy for implementation effort. I agree. So I think you can just ignore RubyVM module.

#13 - 04/07/2019 10:31 AM - Eregon (Benoit Daloze)

mame (Yusuke Endoh) wrote:

> If you think so, you have to implement the feature in any way, regardles of whether it is renamed to CRuby or not? I don't see at all what the renaming will solve.

I think the rename will solve the problem of making it clear, wherever RubyVM is currently used, that it exists only on MRI, and is meant as a CRuby playground, rather than being some portable VM module between Ruby implementations.
It would also make it consistent with the JRuby, TruffleRuby and Rubinius modules.

> I believe that they can be implemented as compatibility layers. For example, TruffleRuby's RubyVM::InstructionSequence can return a MRI-compatible bytecode, but TruffleRuby will not use the bytecode for execution. I think that such a dummy API would be actually useful for some cases, e.g., MRI bytecode-level static analyzer.
>
> You may think it is not worthy for implementation effort. I agree.

Right, I guess it is theoretically possible but I think it wouldn't make much sense (using MRI for a MRI bytecode-level static analyzer makes more sense).
We could just have a dummy implementation returning a RubyVM::InstructionSequence instance with just the code as a String inside, since TruffleRuby cannot expose any bytecode, but that would probably not work for many usages of RubyVM::InstructionSequence, and so I think it doesn't make much sense either.

> So I think you can just ignore RubyVM module.

In some sense, this is what I want, make it clear to everyone that RubyVM functionality is specific to MRI and will not be implemented on other Ruby implementations.
As a consequence, using RubyVM without a fallback is equivalent to saying "this code can only run on MRI and will not run on other Ruby implementations, until such a fallback is provided or RubyVM is no longer used".
I think making it clear requires a rename.
Without a rename, it's not clear. For instance, we just discussed in this issue and I think we now agree RubyVM is not portable but it wasn't clear like that in the beginning.
And that's Ruby implementers discussing, how about users? How should they know RubyVM is MRI-specific?

I think if RubyVM was renamed to CRuby, YARV or MRI, it would be crystal clear this is all MRI-specific.

One way to make it a bit clearer for users is improving the documentation of RubyVM to make it clear everything under it is MRI-specific, and no other Ruby implementation defines that module.
I will make a patch for that, but I think renaming the module is still the best way to communicate this.
It also gives a chance to existing users of RubyVM to realize their code is MRI-specific.

#14 - 04/07/2019 10:51 AM - Eregon (Benoit Daloze)

I just noticed RubyVM.resolve_feature_path.
This is a portable API, and as such should not be mixed with MRI-specific APIs (by being under RubyVM).

This shows the need for a common namespace for experimenting with portable experimental features, as mentioned above. Of course, such a namespace should be separate from implementation-specific APIs, so the user can use the same code no matter which Ruby implementation is used.

An alternative is of course moving resolve_feature_path to a stable API, e.g., as a singleton method of Kernel.
But I think it is good to have a way to have portable and experimental APIs, such that other Ruby implementations can implement it too.
To make it clear, RubyVM.resolve_feature_path cannot be implemented in other Ruby implementations, as it would violate the fact that RubyVM should only exist on MRI, and that is of course unfortunate and a contradiction.

#15 - 04/07/2019 01:40 PM - Eregon (Benoit Daloze)

I made a PR on GitHub to discuss the proposed documentation changes, please take a look and review:
https://github.com/ruby/ruby/pull/2113

#16 - 04/07/2019 01:56 PM - Eregon (Benoit Daloze)

- Related to Feature #15752: A dedicated module for experimental features added

#17 - 04/07/2019 02:00 PM - waheedi (Waheed Barghouthi)

- Backport deleted (2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN)

- ruby -v deleted (ruby 2.6.2p47 (2019-03-13 revision 67232) [x86_64-linux])

- Tracker changed from Bug to Feature

Eregon (Benoit Daloze) wrote:

> I made a PR on GitHub to discuss the proposed documentation changes, please take a look and review:
> https://github.com/ruby/ruby/pull/2113

Eregon (Benoit Daloze), I do believe there is a sense in what you are trying to do, but what makes more sense to me is that Matz Ruby was named "Ruby". You can still clear out the confusion without prepending any letters to Ruby, I believe adding a note like the ones you added "This class is MRI specific as it exposes implementation details" would be sufficient to resolve any confusion. I changed this from a bug to a feature.

#18 - 04/07/2019 02:30 PM - Eregon (Benoit Daloze)

waheedi (Waheed Barghouthi) wrote:

> Eregon (Benoit Daloze), I do believe there is a sense in what you are trying to do, but what makes more sense to me is that Matz Ruby was named "Ruby".

I believe "Ruby" refers to the programming language and from that "Ruby VM" is logically *a* "Virtual Machine executing Ruby code".
There is more than one VM that can execute Ruby code, so I think "the Ruby VM" is simply confusing.
Please see my comment about ruby vs Ruby above too.

> You can still clear out the confusion without prepending any letters to Ruby

I'm not prepending letters, it's what matz himself would call this implementation AFAIK. Here is a blog post explaining it:
http://engineering.appfolio.com/appfolio-engineering/2017/12/28/cruby-mri-jruby-rubyspec-rubinius-yarv-a-little-bit-of-ruby-naming

BTW, that blog post makes a good point:
"Ruby 1.8, YARV and MJIT are all CRuby, but they're different generations of tech within CRuby: the old Ruby 1.8 interpreter, then YARV, then MJIT."

> I believe adding a note like the ones you added "This class is MRI specific as it exposes implementation details" would be sufficient to resolve any confusion.

I think that can be made clear enough in the documentation (if people read it), but it's not clear in the calling code.
So people reading the code would still not be clear about whether that code uses MRI-specific features.
And I would guess many people won't notice the change in documentation of RubyVM.

FWIW, Bundler mentions multiple times in man pages "Ruby VM" and "the running Ruby VM" which means "a Ruby implementation VM", so that's clearly another take on what "Ruby VM" means.

#19 - 04/07/2019 02:56 PM - waheedi (Waheed Barghouthi)

Eregon (Benoit Daloze) wrote:

> waheedi (Waheed Barghouthi) wrote:
>
> > Eregon (Benoit Daloze), I do believe there is a sense in what you are trying to do, but what makes more sense to me is that Matz Ruby was named "Ruby".
>
> I believe "Ruby" refers to the programming language and from that "Ruby VM" is logically *a* "Virtual Machine executing Ruby code".
> There is more than one VM that can execute Ruby code, so I think "the Ruby VM" is simply confusing.
> Please see my comment about ruby vs Ruby above too.
>
> > You can still clear out the confusion without prepending any letters to Ruby
>
> I'm not prepending letters, it's what matz himself would call this implementation AFAIK. Here is a blog post explaining it:
> http://engineering.appfolio.com/appfolio-engineering/2017/12/28/cruby-mri-jruby-rubyspec-rubinius-yarv-a-little-bit-of-ruby-naming
>
> BTW, that blog post makes a good point:
> "Ruby 1.8, YARV and MJIT are all CRuby, but they're different generations of tech within CRuby: the old Ruby 1.8 interpreter, then YARV, then MJIT."
>
> > I believe adding a note like the ones you added "This class is MRI specific as it exposes implementation details" would be sufficient to resolve any confusion.
>
> I think that can be made clear enough in the documentation (if people read it), but it's not clear in the calling code.
> So people reading the code would still not be clear about whether that code uses MRI-specific features.
> And I would guess many people won't notice the change in documentation of RubyVM.
>
> FWIW, Bundler mentions multiple times in man pages "Ruby VM" and "the running Ruby VM" which means "a Ruby implementation VM", so that's clearly another take on what "Ruby VM" means.

I totally agree on your point

> There is more than one VM that can execute Ruby code, so I think "the Ruby VM" is simply confusing.

The pull request you just created does not really reflect what you are saying, The RubyVM module only exists on MRI. +RubyVM+ is not defined in

Even if Matz refer to Ruby as CRuby that does not mean he would like to rename it, but i think that to clarify that he is referring to the C Ruby implementation.

I definitely can see the confusion caused by calling it RubyVM but there were no other implementations when it started, and it made total sense at the time, and I believe changing it would not really help more than 2% of the Ruby enthusiasts and they already know about it by now :)

Have a good day

#20 - 04/07/2019 04:59 PM - Eregon (Benoit Daloze)

waheedi (Waheed Barghouthi) wrote:

> The pull request you just created does not really reflect what you are saying, The RubyVM module only exists on MRI. +RubyVM+ is not defined in

I'm just trying to clarify as much as possible the documentation, until the rename is done.
If the module was named CRuby (or MRI or YARV), there wouldn't even be a need for an explanation that it's MRI-specific and not defined on other Ruby implementations :)

> I definitely can see the confusion caused by calling it RubyVM but there were no other implementations when it started, and it made total sense at the time, and I believe changing it would not really help more than 2% of the Ruby enthusiasts and they already know about it by now :)

I think it would help all people knowing about it (including MRI committers) by making it clear that it's only for MRI-specific features and not, e.g., for portable experimental features (#15752).

> Have a good day

Thanks, you too.

#21 - 04/08/2019 09:53 AM - naruse (Yui NARUSE)

As far as I remember, the name "RubyVM" is introduced because the name of the implementation which is called as CRuby or MRI cannot be decided, and also its VM's name is not actually YARV. To rename it to something, it needs to reopen the pandora box.

I think such topic will be just a bikeshed and shouldn't waste time.