Ruby master - Bug #13188

Reinitialize Ruby VM.

02/03/2017 05:58 PM - joedaniels29 (Joe Daniels)

Status:	Rejected		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:	2.0.0-p648	**Backport:**	2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

It appears that by following what appears to be a pretty standard setup procedure:

```
ruby_init_stack(variable_in_this_stack_frame)
ruby_init();
ruby_init_loadpath();

rb_require("enc/encdb");
rb_require("enc/trans/transdb");
ruby_process_options(Int32(options.count), &cargs)

var state: Int32 = 0;
if ruby_executable_node(node, &state) != 0 {
    state = ruby_exec_node(node)
}
if state != 0 {
    throw RubyError.current
}


ruby_cleanup(state);
```

Its not possible to recreate the VM. Are there instructions on this process? or is this an actionable task for which contributions from a first time contributor would be accepted?

Related issues:

Related to Ruby master - Feature #14792: Multiple RubyVM in one process to ma...	**Feedback**

History

#1 - 02/04/2017 05:43 AM - shyouhei (Shyouhei Urabe)

AFAIK the ruby interpreter uses lots of global variables, which makes it practically impossible to re-initialize.

This is of course not a good thing (antique design). I think mruby is much modern here.

#2 - 02/04/2017 08:19 AM - duerst (Martin Dürst)

Shyouhei Urabe wrote:

> AFAIK the ruby interpreter uses lots of global variables, which makes it practically impossible to re-initialize.

How difficult would it be to fix this?

#3 - 02/06/2017 01:10 AM - shyouhei (Shyouhei Urabe)

Martin Dürst wrote:

> Shyouhei Urabe wrote:
>
> > AFAIK the ruby interpreter uses lots of global variables, which makes it practically impossible to re-initialize.
>
> How difficult would it be to fix this?

Nobu and myself each once tried to move those global variables into the VM struct, to make it possible to have multiple VMs at once (mvm). My try changed hundreds of thousands of lines and resulted in inferior performance. It was because accessing global variable is in fact much faster than to indirectly refer them via VM-stored pointers.

So it is not only very hard to fix, but even when we do so, we have to live with slowness.

#4 - 02/06/2017 02:41 AM - ko1 (Koichi Sasada)

On 2017/02/06 10:10, shyouhei@ruby-lang.org wrote:

> VM-stored pointers

More correctly, thread-local variables (on pthread).

--
// SASADA Koichi at atdot dot net

#5 - 02/07/2017 10:21 PM - normalperson (Eric Wong)

SASADA Koichi ko1@atdot.net wrote:

> On 2017/02/06 10:10, shyouhei@ruby-lang.org wrote:
>
> > VM-stored pointers
>
> More correctly, thread-local variables (on pthread).

Was it function call overhead from pthread_getspecific?

Did you try __thread?

I think __thread was GCC-specific, but clang supports it, too;
and we can fall back to existing pthread_getspecific for other
compilers. I think something similar was introduced for C11,
too, but we're still on C89...

But yeah, having VM struct passed with every function call
(like mrb_state in mruby) is probably most portable and fast.

#6 - 02/08/2017 12:05 AM - ko1 (Koichi Sasada)

On 2017/02/08 7:18, Eric Wong wrote:

> But yeah, having VM struct passed with every function call
> (like mrb_state in mruby) is probably most portable and fast.

Yes. Ruby 3 will use it.

--
// SASADA Koichi at atdot dot net

#7 - 02/08/2017 12:41 AM - normalperson (Eric Wong)

SASADA Koichi ko1@atdot.net wrote:

> On 2017/02/08 7:18, Eric Wong wrote:
>
> > But yeah, having VM struct passed with every function call
> > (like mrb_state in mruby) is probably most portable and fast.
>
> Yes. Ruby 3 will use it.

Good, but we will still keep a compatibility API for C extensions?

#8 - 02/10/2017 11:21 PM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> SASADA Koichi ko1@atdot.net wrote:
>
> > On 2017/02/08 7:18, Eric Wong wrote:
> >
> > > But yeah, having VM struct passed with every function call
> > > (like mrb_state in mruby) is probably most portable and fast.
> >
> > Yes. Ruby 3 will use it.
>
> Good, but we will still keep a compatibility API for C extensions?

Also, can we introducing it in 2.5?
(keeping compatibility API, of course).

#9 - 04/17/2017 05:39 AM - shyouhei (Shyouhei Urabe)

- Assignee set to ko1 (Koichi Sasada)

- Status changed from Open to Assigned

#10 - 05/19/2017 03:05 AM - ko1 (Koichi Sasada)

- Status changed from Assigned to Closed

I close this ticket because we need long-time effort.

> Also, can we introducing it in 2.5?
> (keeping compatibility API, of course).

If we can.

#11 - 06/29/2017 04:37 PM - usa (Usaku NAKAMURA)

- Status changed from Closed to Rejected

#12 - 05/29/2018 05:23 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #14792: Multiple RubyVM in one process to make real multi-threading. added