Ruby trunk - Feature #3187

Allow dynamic Fiber stack size

04/23/2010 02:46 AM - mperham (Mike Perham)

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

Description

=begin
I'd like a way to increase the size of the Fiber stack dynamically so when my program starts, I can set it to whatever value I need for the code I'm running. 4KB is too easy to run into problems when running recursive code but settling on any arbitrary static value seems pointless to me.
=end

Related issues:

Related to Ruby trunk - Feature #6694: Thread.new without block. **Assigned**

History

#1 - 04/23/2010 02:48 AM - mperham (Mike Perham)

=begin
To be clear, I'm not asking to extend the stack of a currently executing Fiber, just a one-time configuration when my program starts. Something like this:

```
require 'fiber'
Fiber.stack_size = 16 * 1024

Fiber.new do
  parse_some_big_xml
end.resume
```

=end

#2 - 04/23/2010 02:55 AM - mame (Yusuke Endoh)

- Target version set to 2.0.0

=begin

=end

#3 - 10/12/2011 09:42 PM - rupert (Robert Pankowecki)

I would also welcome such improvement.

#4 - 10/19/2011 11:26 PM - nagachika (Tomoyuki Chikanaga)

- File fiber_stacksize.patch added

- Category set to core

Hi,

I've written a patch. This patch adds some methods Fiber.default_vm_stacksize, Fiber.default_vm_stacksize=, Fiber#vm_stacksize and add an optional Hash argument to Fiber#initialize, and tests for them.
You can specify default VM stack size of Fiber created afterward and/or specify individually when create a Fiber.

ex)
Fiber.default_vm_stacksize = 16 * 1024

Fiber.new do
do_something
end

or

```
Fiber.new(:vm_stacksize => 16 * 1024) do
do_simething
end
```

Note that the size of VM stack is number of Objects and memsize of stack is vm_stacksize * sizeof(VALUE) bytes.
Also note that this patch enable to change only VM stack size, but not machine stack size. I think when Fiber is implemented based on makecontext/swapcontext (FIBER_USE_NATIVE=1), machine stack size (default: 64KB) can be configurable. I wonder if procedures like `parse_some_big_xml' in Mike's example also need larger machine stack size? Does anyone have such a testcase?

#5 - 10/20/2011 03:53 AM - kosaki (Motohiro KOSAKI)

> Note that the size of VM stack is number of Objects and memsize of stack is vm_stacksize * sizeof(VALUE) bytes.
> Also note that this patch enable to change only VM stack size, but not machine stack size. I think when Fiber is implemented based on makecontext/swapcontext (FIBER_USE_NATIVE

#6 - 10/20/2011 01:01 PM - nagachika (Tomoyuki Chikanaga)

> I don't think we should export vm_stack. It's purely implementation
> detail. Just expose "stack size"
> knob and it should change both vm-stack and machine-stack size.
> You're right. It's better to provide more abstract way to tune memory usage of Fiber.
> How about like the following?

```
Fiber.stacksize = 2.0  # => twice the size of default stack size
```

or maybe we have to examine how VM/machine stack are consumed.

And even tough we provide such a unified interface, I think more low-level,
implementation specific methods are convenient when users need fine-tuned parameter set for
their applications. Is it a bad idea?
for example, ObjectSpace.count_objects etc.. tightly depend on MRI implementation,
and are useful just because of it.

#7 - 10/24/2011 08:23 AM - spatulasnout (B Kelly)

Tomoyuki Chikanaga wrote:

> I've written a patch. This patch adds some methods Fiber.default_vm_stacksize, Fiber.default_vm_stacksize=, Fiber#vm_stacksize and add an optional Hash argument to Fiber#initialize, and tests for them.
> You can specify default VM stack size of Fiber created afterward and/or specify individually when create a Fiber.
>
> ex)
> Fiber.default_vm_stacksize = 16 * 1024
>
> Fiber.new do
> do_something
> end
>
> or
>
> Fiber.new(:vm_stacksize => 16 * 1024) do
> do_simething
> end

Very nice!

My application uses fibers extensively, and it began to exceed the
default fiber stack during recursive traversal of relatively shallow
trees. (Not surprising, given the 4K default stack size.)

I had patched cont.c locally as follows:

```
#define FIBER_STACK_SIZE_SCALE  8  /* need more fiber stack space */

#define FIBER_MACHINE_STACK_ALLOCATION_SIZE  (0x10000 *
FIBER_STACK_SIZE_SCALE)
```

#define FIBER_VM_STACK_SIZE ((4 * 1024) * FIBER_STACK_SIZE_SCALE)

> Note that the size of VM stack is number of Objects and memsize of stack is vm_stacksize * sizeof(VALUE) bytes.
> Also note that this patch enable to change only VM stack size, but not machine stack size. I think when Fiber is implemented based on makecontext/swapcontext (FIBER_USE_NATIVE=1), machine stack size (default: 64KB) can be configurable. I wonder if procedures like `parse_some_big_xml' in Mike's example also need larger machine stack size? Does anyone have such a testcase?

Can anyone comment on the relationship between the VM stack size and the
machine stack size?

I scaled them both equally to be "safe".

But I don't know how they are related, so I'm not sure if the
corresponding increase to the machine stack was necessary?

Thanks,

Bill

#8 - 10/31/2011 01:23 AM - ko1 (Koichi Sasada)

Hi,

I agree with this proposal. Also add same parameter setting feature for
Thread.

However, I can't make good API to specify stack (VM and machine) size
(and other parameters if there are).

There are several proposals.

Type 1: Thread creating argument (ex: Thread.new(stack_size: 1024))
Type 2: Thread global parameter (ex: Thread.stack_size = 1024)

I think Type 2 is not good (to set default value is okay. However it
conflicts other usage. Typically thread-unsafe).

I also think Type 1 has also problem. If you want to pass keyword
argument "stack_size" to fiber or thread, it should be conflict.

My idea is creating new Thread (Fiber) class with new parameter:

```
  MyThread = Thread.new_template(stack_size: 1024)
  MyThread.new{
    ...
  }
```

`# Of course "new_template" is bad name.`

Any ideas?

--
// SASADA Koichi at atdot dot net

#9 - 10/31/2011 07:53 AM - headius (Charles Nutter)

Quick comments.

- JVM can specify per-thread stack size but always in bytes. Any API that exposes stack size should be abstract rather than require a knowledge of stack frame sizes from impl to impl. The "stack size factor" version is pretty good.

- I agree it would be nice to specify per-thread stack sizes too. We must do this for JRuby on Android, and have to use JRuby- specific mechanisms for it.

- Charlie (mobile)

#10 - 03/18/2012 02:36 PM - nahi (Hiroshi Nakamura)

- Assignee set to ko1 (Koichi Sasada)

How about;

```
t = Thread.new(stack_size: 1024)
t.stack_size = 1024
t[:initial_tls_hash] = {}
t.run {
  ...
}
```

#11 - 03/18/2012 06:46 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#12 - 06/26/2012 04:20 AM - ko1 (Koichi Sasada)

Any other idea about it? It's only API design.

I'm not sure nahi-san's idea is good for Ruby or not.
I feel that it is too different from current style.

#13 - 07/04/2012 01:59 PM - ko1 (Koichi Sasada)

I make another ticket about it:
https://bugs.ruby-lang.org/issues/6694

Thanks,
Koichi

--
// SASADA Koichi at atdot dot net

#14 - 10/27/2012 05:37 AM - ko1 (Koichi Sasada)

I'm considering it because no progress on https://bugs.ruby-lang.org/issues/6694 (sorry, it is naming issue, I think).

Can I add an environment variable (such as RUBY_FIBER_MACHINE_STACK_SIZE) to avoid this issue temporarily?

#15 - 11/24/2012 12:07 PM - mame (Yusuke Endoh)

- Target version changed from 2.0.0 to 2.6

#16 - 09/30/2013 08:16 PM - ko1 (Koichi Sasada)

- Status changed from Assigned to Feedback

Ruby 2.0 already has
RUBY_VM_FIBER_VM_STACK_SIZE
RUBY_FIBER_MACHINE_STACK_SIZE

is it enough?

#17 - 01/12/2015 06:23 PM - jaredbeck (Jared Beck)

```
Ruby 2.0 already has
RUBY_VM_FIBER_VM_STACK_SIZE
RUBY_FIBER_MACHINE_STACK_SIZE
```

Are these environment variables to configure the stack size? Is there documentation on usage? (I mean, what are the units? bytes, kb?) Is there a way to read the default value programmatically in ruby?

#18 - 05/31/2016 04:28 PM - headius (Charles Nutter)

FYI, it appears at least one Ruby implementation has implemented this unilaterally:
https://github.com/rubinius/rubinius/commit/c26139a03132661202f30c778ac9e7bc489959d4

We'd also like to support this feature in JRuby, but we'd prefer to go through official channels.

I do not believe env vars are sufficient because you may have different libraries that want to tune their threads/fibers to different sizes.

#19 - 07/20/2016 01:57 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #6694: Thread.new without block. added

#20 - 07/20/2016 02:03 AM - shyouhei (Shyouhei Urabe)

We looked at this issue at yesterday's developer meeting.

The (potential) problem here is the way Rubinius implements stack size accidentally breaks compatibility of how Thread.new works. In the current API all the argument passed to this method are forwarded to its block. As far as I read the patch pointed by Charles it seems Rubinius chose to break here and let it eat the keyword arguments. This might work -- given Rubinius lives without any serious problem around it -- but does change the way it works in the MRI.

Another approach would be to separate a thread creation; then inject arguments to it; then finally kick it to run. This is what ko1 proposed in issue #6694.

#21 - 02/21/2018 02:41 PM - jjyr (Jinyang Jiang)

How about

```
Thread.with_configure(stack_size: 1024).new(a: 1, b: 2){}
# or
Thread::Config.new(stack_size: 1024).start(a: 1, b: 2){}
# alias start new
```

ko1 (Koichi Sasada) wrote:

> Hi,
>
> I agree with this proposal. Also add same parameter setting feature for
> Thread.
>
> However, I can't make good API to specify stack (VM and machine) size
> (and other parameters if there are).
>
> There are several proposals.
>
> Type 1: Thread creating argument (ex: Thread.new(stack_size: 1024))
> Type 2: Thread global parameter (ex: Thread.stack_size = 1024)
>
> I think Type 2 is not good (to set default value is okay. However it
> conflicts other usage. Typically thread-unsafe).
>
> I also think Type 1 has also problem. If you want to pass keyword
> argument "stack_size" to fiber or thread, it should be conflict.
>
> My idea is creating new Thread (Fiber) class with new parameter:
>
> ```
>   MyThread = Thread.new_template(stack_size: 1024)
>   MyThread.new{
>     ...
>   }
> ```
>
> ```
> # Of course "new_template" is bad name.
> ```
>
> Any ideas?
>
> --
> // SASADA Koichi at atdot dot net

Files

fiber_stacksize.patch	5.89 KB	10/19/2011	nagachika (Tomoyuki Chikanaga)