Ruby - Feature #6694

Thread.new without block.

07/04/2012 01:51 PM - ko1 (Koichi Sasada)

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

Description

Abstract

Support Thread.new() without block.

Before: Thread.new(params...){|thread_local_params| ...}

After: Thread.new(proc: lambda{|tl_params...| ...}, args: params..., other_thread_config...)

Background

Thread.new creates new Thread object and run passed block in another thread immediately. Thread.new can receive parameters and pass all parameters to block.

```
Thread.new(a, b, c) do |ta, tb, tc|
  # ta, tb, tc is thread local
}
```

There are some request to specify thread configurations such as stack size described in [Ruby 1.9 - Feature #3187] (in this case, stack size for Fiber.new). However, we have no way to pass such thread configuration on the Thread.new().

Proposal

Allow Thread.new() without block. A block will be passed with proc parameter. Passed arguments will be passed with args parameter.

```
# ex1
Thread.new(){...}
#=>
Thread.new(proc: -> {...})

# ex2
Thread.new(a, b, c){|ta, tb, tc| ...}
#=>
Thread.new(proc: ->(ta, tb, tc){ ... }, params: [a, b, c])
```

If you want to specify stack size, then:

Thread.new(stack_size: 4096, proc: proc{...}, args: [a, b, c])

Note that I'll make another ticket for thread (and fiber) creation parameters.

This change can be described with the following pseudo code:

```
def Thread.new(*args, &block)
  if block
    Thread.new_orig(*args, &block)
  else
    config = args[0] || raise ArgumentError
    stack_size = config[:stack_size]
    # ... and process another parameters
    Thread.new_orig(*config[:args], &config[:proc])
  end
```

```
end
```

Another proposal

On the [ruby-core:43385], Nahi-san proposed that if no block given on Thread.new(), then create "waiting" thread. Thread#run kicks waiting thread with parameters.

```
th = Thread.new(thread_config_params)
...
th.run(params){|thread_local_params|
  ...
}
```

We can combine with proc: parameter and this proposal.
If Thread.new() doesn't have block and proc: parameter, then making a waiting thread.

NOTE: Because we have already Thread#run, Thread#start is better than Thread#run?

Note

I don't make any survey on other languages. Please give us your comments.

Related issues:

Related to Ruby - Feature #11251: Thread#name and Thread#name=	**Closed**
Related to Ruby - Feature #3187: Allow dynamic Fiber stack size	**Rejected**

History

#1 - 07/04/2012 01:51 PM - ko1 (Koichi Sasada)

- Tracker changed from Bug to Feature

#2 - 07/14/2012 06:40 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

#3 - 07/20/2012 03:32 PM - brixen (Brian Shirai)

I object to this API for at least the two following reasons:

1. Stack size is an implementation detail and coupling Ruby code to details of a particular implementation is undesirable. Applications may be developed on one implementation and deployed on another. Or details affecting stack size may change between versions of a single implementation.
2. Stack size may depend on code not in the application or library (eg a library using Thread.new that calls application code or application code that different versions or implementations of a library).

This setting should be a configuration option, not a Ruby method API.

Cheers,
Brian

#4 - 07/20/2012 03:46 PM - shyouhei (Shyouhei Urabe)

Forgot to mention to @_ko1 that both POSIX and Windows API starts threads immediately when they are created.

It is not that obvious what should happen when a thread that was created is not running. For instance, to join that thread should do what?

#5 - 07/20/2012 03:59 PM - ko1 (Koichi Sasada)

(2012/07/20 15:32), brixen (Brian Ford) wrote:

> I object to this API for at least the two following reasons:
>
> 1. Stack size is an implementation detail and coupling Ruby code to details of a particular implementation is undesirable. Applications may be developed on one implementation and deployed on another. Or details affecting stack size may change between versions of a single implementation.
> 2. Stack size may depend on code not in the application or library (eg a library using Thread.new that calls application code or application code that different versions or implementations of a library).
>
> This setting should be a configuration option, not a Ruby method API.

Maybe you mention to this ticket, don't you?
https://bugs.ruby-lang.org/issues/6695

This ticket only propose thread configuration before thread creation.

--
// SASADA Koichi at atdot dot net

#6 - 07/20/2012 03:59 PM - ko1 (Koichi Sasada)

(2012/07/20 15:55), SASADA Koichi wrote:

> This ticket only propose thread configuration before thread creation.

Sorry, "configurable thread before running".

--
// SASADA Koichi at atdot dot net

#7 - 07/20/2012 04:23 PM - ko1 (Koichi Sasada)

(2012/07/20 15:46), shyouhei (Shyouhei Urabe) wrote:

> Forgot to mention to @_ko1 that both POSIX and Windows API starts threads immediately when they are created.
>
> It is not that obvious what should happen when a thread that was created is not running.

On POSIX and Windows threads, the creation API has configuration
parameter. Thread.new doesn't have.

> For instance, to join that thread should do what?

Two possible solutions.

1. raise ThreadError
2. wait for thread run and exit.

On second solution, it is easy to understand that the thread created by
Thread.new without procedure (block or parameter) start and stop
(Thread#wait) immediately.

But I prefer 1.

--
// SASADA Koichi at atdot dot net

#8 - 07/20/2012 05:26 PM - shyouhei (Shyouhei Urabe)

ko1 (Koichi Sasada) wrote:

> (2012/07/20 15:46), shyouhei (Shyouhei Urabe) wrote:
>
>> Forgot to mention to @_ko1 that both POSIX and Windows API starts threads immediately when they are created.
>>
>> It is not that obvious what should happen when a thread that was created is not running.
>
> On POSIX and Windows threads, the creation API has configuration
> parameter. Thread.new doesn't have.

So?

I'm not against adding a new thread constructor.

The problem is that new one do not start.

> But I prefer 1.

What happens when Thread.kill called with that thread?

What happens when Thread#backtrace was called?
What if Thread#status?

You have to define all those operations. That is what I call "not obvious".

#9 - 07/20/2012 05:29 PM - shyouhei (Shyouhei Urabe)

Also, I'd like to let you know that pthread_create() comes with their design decisions as RATIONALE.
http://pubs.opengroup.org/onlinepubs/009695399/functions/pthread_create.html

Please read it.

#10 - 07/20/2012 10:42 PM - brixen (Brian Shirai)

ko1 (Koichi Sasada) wrote:

> (2012/07/20 15:32), brixen (Brian Ford) wrote:
>
>> I object to this API for at least the two following reasons:
>>
>> 1. Stack size is an implementation detail and coupling Ruby code to details of a particular implementation is undesirable. Applications may be developed on one implementation and deployed on another. Or details affecting stack size may change between versions of a single implementation.
>> 2. Stack size may depend on code not in the application or library (eg a library using Thread.new that calls application code or application code that different versions or implementations of a library).
>>
>> This setting should be a configuration option, not a Ruby method API.
>
> Maybe you mention to this ticket, don't you?
> https://bugs.ruby-lang.org/issues/6695
>
> This ticket only propose thread configuration before thread creation.

I don't think I understand your question. To summarize my objection to this proposed API change: Thread stack size should be something set at the VM level completely outside of Ruby code. Ruby code should not be coupled with implementation details.

Cheers,
Brian

#11 - 07/20/2012 11:53 PM - ko1 (Koichi Sasada)

(2012/07/20 22:42), brixen (Brian Ford) wrote:

> I don't think I understand your question. To summarize my objection to this proposed API change: Thread stack size should be something set at the VM level completely outside of Ruby code. Ruby code should not be coupled with implementation details.

I think I understand your question. And your question should be
discussed on ticket 6695 https://bugs.ruby-lang.org/issues/6695.

My proposal is *not* adding "initial stack size parameter for thread
creation".
It is only "configurable thread parameters". More general framework.

I propose "stack size parameter" at next ticket (using this ticket):
https://bugs.ruby-lang.org/issues/6695

This ticket (6694):
Change Thread.new() API to accept any initial parameters.

Next ticket (6695):
New parameters including stack size (machine stack size and VM
stack size).
Not only stack size, but also "name" and others.
I think "name" is killer usage of this ticket (6694).

--
// SASADA Koichi at atdot dot net

#12 - 07/21/2012 01:27 AM - brixen (Brian Shirai)

ko1 (Koichi Sasada) wrote:

> (2012/07/20 22:42), brixen (Brian Ford) wrote:
>
> > I don't think I understand your question. To summarize my objection to this proposed API change: Thread stack size should be something set at the VM level completely outside of Ruby code. Ruby code should not be coupled with implementation details.
>
> I think I understand your question. And your question should be
> discussed on ticket 6695 https://bugs.ruby-lang.org/issues/6695.
>
> My proposal is *not* adding "initial stack size parameter for thread
> creation".
> It is only "configurable thread parameters". More general framework.
>
> I propose "stack size parameter" at next ticket (using this ticket):
> https://bugs.ruby-lang.org/issues/6695
>
> This ticket (6694):
> Change Thread.new() API to accept any initial parameters.
>
> Next ticket (6695):
> New parameters including stack size (machine stack size and VM
> stack size).
> Not only stack size, but also "name" and others.
> I think "name" is killer usage of this ticket (6694).

Thread#name doesn't require a new API for Thread.new. Thread#name makes sense as something that can be set any time.

The pthread_create() design rationale referenced by Urabe-san is relevant to this proposed API. For the same reasons discussed there, I would object to this API change. I don't see any real need for this API if stack size is not one of the things being set. Hence, this API proposal really looks like just a thin rationale for that proposal.

Cheers,
Brian

#13 - 07/21/2012 06:51 AM - headius (Charles Nutter)

I will comment on stack sizing on the other issue.

As far as passing params and not starting the thread, it doesn't seem like a bad API. There are other thread characteristics that could be passed at construction time like abort_on_exception, priority, and potentially things like "start" (to autostart as the current API), "daemon" to create threads that will keep the VM alive after main thread exits, and so on.

FWIW, JVM threads do not run until explicitly started. The start of the actual native thread is deferred until Thread#start is called. Thread-runtime operations raise exceptions if called before the thread is started. Simple enough.

#14 - 07/21/2012 02:53 PM - kosaki (Motohiro KOSAKI)

> I don't think I understand your question. To summarize my objection to this proposed API change: Thread stack size should be something set at the VM level completely outside of Ruby code. Ruby code should not be coupled with implementation details.

If I understand correctly, you suggested Thread and Fiber should have
enough large stack size. I 100% agree. Then almost all ruby
programmers don't need bother stack size issue.

The problem is, *current* fiber implementation have merely very small
stack size and it often caused stack overflow issue. Why it is so
small? Because of, larger stack size restricted maximum number of
fibers. Think, stack-size * number-of-fibers should be < 3G if system
is running on 32bit OS.

So, we hope to increase stack size for just your opinion. but it has
one down size. it reduce a number of maximum fibers. The next question
is, anybody need such so many thread/fibers? Is this real issue?

Unfortunately, we can't answer it. It depend on ruby scripts. Out of
ruby world, I know some game programmer prefer to create a lot of
fibers. But I don't know real ruby use case. However, I hope to keep
a way to create a lot of fibers if programmer don't hesitate deep
system/cpu depending scripts.

Finally, if you agree fiber issue and you think only fiber should have

stack size parameter, can you please explain why do you dislike thread
and fiber have the same interface? Usually, thread and fiber should
keep same or similar interface because it reduce learning cost.

Thank you.

#15 - 07/25/2012 07:02 PM - ko1 (Koichi Sasada)

Some developers proposed that other method names like: Thread.conf_new should be introduced to specify per thread parameters.

Because Thread.new(...) without block causes miss like:

```
Thread.new(name: 'foo'){
}
```

In this case, {name: 'foo'} will be passed as block parameter for thread and it doesn't make any warnings.

I'm not sure the name 'Thread.conf_new()' is good name or not.
Thread.new_configured(...) is too long?

# Thread.new2 is good joke.

#16 - 10/27/2012 07:15 AM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 2.6

I don't have good name about it.
I postpone this ticket.

#17 - 06/12/2015 06:20 AM - naruse (Yui NARUSE)

- Related to Feature #11251: Thread#name and Thread#name= added

#18 - 07/20/2016 01:57 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #3187: Allow dynamic Fiber stack size added

#19 - 01/31/2017 08:08 AM - ko1 (Koichi Sasada)

- Description updated

#20 - 01/31/2017 08:32 AM - ko1 (Koichi Sasada)

Existing Ideas:

- (1) allow keywords for new() (like new(name: "worker-thread"))
- (1-1) introducing it immediately and break compatibility
- (1-2) introducing it and introduce proper keyword semantics (I doubt we can close this discussion)
- (2) create threads by new() without block
- (2-1) provide proc: keyword [Feature #6694]
- (2-2) suspend thread and configure it before running https://bugs.ruby-lang.org/issues/3187#note-10
- (3) Make builders (templates) with specific parameters https://bugs.ruby-lang.org/issues/3187#note-8

Simple one is (1-1), but it can introduce compatibility issue.
Of course, I'm not sure how affect this incompatibility, but csearch Thread.new\\\(| grep 'name:' doesn't match on all of gems.
(name: is only one property so it is not right example)

#21 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)