Ruby master - Feature #6695

Configuration for Thread/Fiber creation

07/04/2012 02:24 PM - ko1 (Koichi Sasada)

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

Description

=begin
= Abstract

With Feature #6694, the following configuration parameters should be allowed for Thread/Fiber creation.

Group1 (new parameters):

- name: Thread/Fiber name
- vm_stack_size: VM's stack size
- machine_stack_size: Machine stack size

Group2 (existing parameters):

- local_storage: Initial Thread/Fiber local parameters
- thread_group: Thread group (Thread only)
- priority: Initial priority Thread#priority= (Thread only)
- abort_on_exception: abort on exception (Thread only)

= Background

With Feature #6694, we have a way to specify configurations for Thread creation. Fiber.new() don't receive any parameters now.

= Proposal

This is a initial proposal of configuration for Thread/Fiber creation.

Group1 (new parameters):

- name: Thread/Fiber name
- vm_stack_size: VM's stack size
- machine_stack_size: Machine stack size

vm_stack_size and machine_stack_size are OS dependent (This means that it will be *hint* parameter).

Thread#inspect should use `name' parameter.

I also propose a new method Thread#name to get the thread name specified by this parameter.

Group2 (existing parameters):

- local_storage: Initial Thread/Fiber local parameters
- thread_group: Thread group (Thread only)
- priority: Initial priority Thread#priority= (Thread only)
- abort_on_exception: abort on exception (Thread only)

Now, we can specify Group2 parameters only *after* thread creation. With this parameter, we can specify parameters before thread creation.

=end

Related issues:

Related to Ruby master - Feature #11251: Thread#name and Thread#name=	**Closed**

History

#1 - 07/04/2012 02:26 PM - ko1 (Koichi Sasada)

- Description updated

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

- Status changed from Open to Assigned

#3 - 07/21/2012 01:23 AM - brixen (Brian Shirai)

I object to this API for setting stack size from Ruby 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 VM configuration option, not a Ruby method API.

Thread#name doesn't need a new Thread.new API.

Cheers,
Brian

#4 - 07/21/2012 02:59 AM - ko1 (Koichi Sasada)

(2012/07/21 1:23), brixen (Brian Ford) wrote:

> I object to this API for setting stack size from Ruby 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 VM configuration option, not a Ruby method API.

Thank you for your comment.

This issue was started from Feature #3187 "Allow dynamic Fiber stack
size" by mperham (sorry, I had needed to show it).

mperham wants to specify fiber size per fiber because current fiber's
default size is too small to do complicated procedures and no way to
change this size.

Current default fiber's machine stack is small because of two reasons.

1. On the 32bit CPU, there are address space restriction and Only a few thousand of fibers (and threads) can be made if we allocate 1MB per fibers. On the Fiber's usage, it is too small in some cases.
2. 1MB memory allocation hit performance in little. (It is very weak reason for Rubyist)

To specify stack size, there are several way.

(1) VM configuration option (as Brian say)
(2) Global config by method (Thread.stack_size=) (as mperham proposed)
(3) Per-thread configuration (as I proposed)

I don't make any objection. It is acceptable.

However, we can't mix "small" and "large" size stack size in same
program. (2) is not perfect because it is not thread-safe.

BTW, the following API is thread-safe. Is it acceptable?

```
Thread.stack_size(size){
# only in this block, the stack size will be `size'
Thread.new(...){
...
}
}
```

On the #3187 thread, nagachika proposed more high level specifier
"factor" >https://bugs.ruby-lang.org/issues/3187#note-6>

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

I think using factor is also acceptable (the method name should be
changed, I guess). But we need to specify.

I also think more rough specifiers such as :small, :medium, :large are
also acceptable.

Fiber.new(size: :small){...}

(for example, at VM configuration with each sizes,
ruby -Xsmall_ss=4KB -Xmedium_ss=128KB -Xlarge_ss=1MB script
)

I don't care low-level or high-level specifier. But I think we need per
thread/fiber specifier.

From developer's perspective: We will increase default stack size for
Fiber (maybe same as Thread) after we introduce this feature. Most
case, larger stack size doesn't make problem (small stack size makes
problems such as mperham's situation).

If application or library needs huge number of Fibers, then this will
feature help.

> Thread#name doesn't need a new Thread.new API.

Yes. Thread#name and Thread#name= is enough if specify a name after
creation like:

Thread.new{
Thread.current.name = 'foo' # Thread.name may be also okay
...
}

But I want to specify a name at or before thread creation.
Is it only for me?

--
// SASADA Koichi at atdot dot net

#5 - 07/21/2012 07:02 AM - headius (Charles Nutter)

Stack size:

I will agree in part with Brian's statement about stack size being too implementation-dependent. It's also 32/64-bit dependent. On the JVM, it's also platform-dependent (Windows will have different stack layout than Linux). In addition, the size of frames on the stack can vary even within the same runtime as code gets JIT compiled and condenses multiple calls' frames into one.

So I don't have a strong objection to being able to specify stack size (java.lang.Thread allows this) but it won't be possible to select a single size that works for all implementations (unless it's large enough no implementation would hit it). This also applies to "small", "medium", "large" and ratios...the JVM itself chooses stack size per-platform, so the actual stack sizes will vary and stack size ratios/tiers still won't produce consistent results across platforms or impls.

For a specific runtime on a specific platform, specifying stack size may be useful (e.g. Ruboto (JRuby on Android) does it for threads on Dalvik, where stack depth is often too small for our interpreter).

Name:

I don't see a reason *not* to support :name in config, if pre-run config is added. #name and #name= probably could come along...JVM threads support setting name, and I've always thought of this as a gap in Thread's API.

#6 - 10/27/2012 07:16 AM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 2.6

Same as #6693 ...

#7 - 06/12/2015 06:20 AM - naruse (Yui NARUSE)

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

#8 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)