Ruby - Feature #4649

Adding parallel constructors to Ruby 2.0

05/05/2011 09:19 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Status:	Rejected	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

I was not sure if this was RubySpec or Feature type.

My request is to create some new syntax for easing the write of concurrent code with Ruby. The syntax could be something like this:

```
parallels do
task "get response from service A" do # optional description
a = IO.read("http://serviceA/request")
end
task { b = get_response_from_service_b }
end
```

at this point both tasks have finish, it's like a join in the threads at the end of the parallel block.

```
call_another_service_that_depends_on(a and b)
```

I am not sure, though, how to deal with scopes inside the parallel block. Traditionally in Ruby, this wouldn't work because both 'a' and 'b' would be local to the 'task' block and would not be accessible from the outside of the parallel block. If we don't want instance variables, I don't know what is the best approach for avoiding some "a=nil; b=nil" before the parallel block.

Maybe this could start as a gem, but having it implemented directly in Ruby would be awesome, specially for scripts that usually don't rely in external dependencies for simplifying distribution...

Any thoughts about this propose?

History

#1 - 05/06/2011 02:17 AM - vjoel (Joel VanderWerf)

IMO this is not really about parallel computation, but about syntax and scope, and it can be solved without changing ruby.

One solution that seems fairly idiomatic to me is:

```
def get_response_from_service_a; "a response"; end
def get_response_from_service_b; "b response"; end

class Parallels
def initialize
@tasks = []
end
def task &task
@tasks << Thread.new(&task)
end
def values
@tasks.map {|task| task.value}
end
end

def parallels &task_block
par = Parallels.new
par.instance_eval &task_block
par.values
end

a, b = parallels do
task { get_response_from_service_a }
task { get_response_from_service_b }
end

p a, b
```

#2 - 05/06/2011 05:53 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

You're right Joel. Maybe the syntax I'm requesting to be included in standard Ruby should be something like:

```
concurrently do
task {...}
...
end
```

But, although the implementation is simple, having a standard idiom for that common requirement would be great. That means I wouldn't have to redefine this idiom not to rely on external gem while doing some scripting... Other than that, making it a language feature could allow some low level improvements in performance, but it is just a guess since I don't know how this could be achieved yet.

Furthermore, depending on the specs, maybe it wouldn't be necessary to do something like "a, b = concurrently do...". Being a standard idiom, someone looking at a code like this would know for sure what is happening since it is a language feature instead of an external dependency...

Also, my initial example was too simple, but I imagine possibly several variables inside each task that should be available at the end of the parallels/concurrently block.

#3 - 05/06/2011 10:02 AM - naruse (Yui NARUSE)

- Tracker changed from Misc to Feature

#4 - 10/18/2011 09:16 AM - naruse (Yui NARUSE)

- Project changed from Ruby to 14

- Target version deleted (3.0)

#5 - 10/23/2011 05:21 PM - naruse (Yui NARUSE)

- Project changed from 14 to Ruby

#6 - 03/18/2012 06:46 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#7 - 06/08/2012 03:22 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

- File feature-4649.odp added

Third wish!

#8 - 06/08/2012 08:41 AM - mame (Yusuke Endoh)

Received.

You want just Kernel#concurrently and Kernel#task?
And, Kernel#concurrently returns an array that contains the result values returned by inside Kernel#tasks. Right?
The array may be shuffled, or in the same order as task definitions?

```
ary = concurrently do
task { 1 }
task { 2 }
task { 3 }
end
p ary #=> [1, 2, 3]
```

--
Yusuke Endoh mame@tsg.ne.jp

#9 - 06/08/2012 10:06 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

If we're going to care about the results of the tasks, which I think it is a good idea, although it was really yours and not mine, it only makes sense to me for the array to have the results in the same order as the tasks were defined, right?

#10 - 07/23/2012 10:19 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Rejected

Rodrigo Rosenfeld Rosas,

Sorry but this proposal was rejected at the developer meeting (7/21).

Please start it as a gem. Matz might reconsider this proposal if
your gem really becomes popular or even de facto.

--
Yusuke Endoh mame@tsg.ne.jp

#11 - 07/24/2012 12:45 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Okay, thanks for considering it :)

Files

feature-4649.odp	18.1 KB	06/08/2012	rosenfeld (Rodrigo Rosenfeld Rosas)