Ruby - Bug #10723

[PERF] bm_tread_create_join 20% slower

01/09/2015 09:38 AM - tgxworld (Guo Xiang Tan)

Status:	Closed		
Priority:	Normal		
Assignee:	akr (Akira Tanaka)		
Target version:			
ruby -v:	ruby 2.2.0dev (2014-09-21 trunk 47676)	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

Relevant commits: https://github.com/ruby/ruby/compare/39fd4a8...5697b2f

Chart showing the regression: http://rubybench.org/ruby/ruby/commits?result_type=vm_thread_create_join

For those unable to view the chart, the benchmark results have increased from 1.74~ seconds to 2.1~ seconds.

Related issues:

Related to Ruby - Bug #10922: TracePoint#binding may return nil in Ruby 2.2	**Closed**

Associated revisions

Revision 8341136f0743199b77e2fd816d625e707b9cd485 - 01/28/2015 05:06 PM - Eric Wong

thread.c: micro-optimize thread create/join

- thread.c (struct join_arg): restructure and make smaller
  (thread_join_sleep): avoid timeofday() call if forever
  (thread_join): pass join_arg.delay directly
  (rb_thread_inspect_msg): remove, inline into rb_thread_inspect
  (rb_thread_inspect): reduce branching and string creation
- thread_pthread.c (native_set_thread_name): create string directly
  to avoid reparsing. [Misc #10723]

This reduces time in benchmark/bm_vm_thread_create_join.rb by
a few percent.

Minor improvements only:

target 0: 2.1.5 (ruby 2.1.5p273 (2014-11-13 revision 48405) [x86_64-linux])
target 1: trunk (ruby 2.3.0dev (2015-01-16 trunk 49282) [x86_64-linux])
target 2: built (ruby 2.3.0dev (2015-01-16 trunk 49282) [x86_64-linux])

benchmark results:
minimum results in each 3 measurements.
Execution time (sec)

name	2.1.5	trunk	built
vm_thread_create_join	1.049	1.242	1.138

Speedup ratio: compare with the result of `2.1.5' (greater is better)

name	trunk	built
vm_thread_create_join	0.845	0.923

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@49430 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 8341136f - 01/28/2015 05:06 PM - Eric Wong

thread.c: micro-optimize thread create/join

- thread.c (struct join_arg): restructure and make smaller
  (thread_join_sleep): avoid timeofday() call if forever
  (thread_join): pass join_arg.delay directly
  (rb_thread_inspect_msg): remove, inline into rb_thread_inspect
  (rb_thread_inspect): reduce branching and string creation
- thread_pthread.c (native_set_thread_name): create string directly
  to avoid reparsing. [Misc #10723]

This reduces time in benchmark/bm_vm_thread_create_join.rb by

a few percent.

Minor improvements only:

target 0: 2.1.5 (ruby 2.1.5p273 (2014-11-13 revision 48405) [x86_64-linux])
target 1: trunk (ruby 2.3.0dev (2015-01-16 trunk 49282) [x86_64-linux])
target 2: built (ruby 2.3.0dev (2015-01-16 trunk 49282) [x86_64-linux])

benchmark results:
minimum results in each 3 measurements.
Execution time (sec)
name 2.1.5 trunk built
vm_thread_create_join 1.049 1.242 1.138

Speedup ratio: compare with the result of `2.1.5' (greater is better)
name trunk built
vm_thread_create_join 0.845 0.923

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@49430 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 01/09/2015 09:40 AM - tgxworld (Guo Xiang Tan)

Opps it should bm_vm_thread_create_join.rb

#2 - 01/16/2015 10:08 AM - normalperson (Eric Wong)

Related, but I do not read Japanese:
https://bugs.ruby-lang.org/issues/10297

#3 - 01/17/2015 03:17 AM - normalperson (Eric Wong)

Minor micro-optimization, I could not find much improvement while
keeping functionality:

http://80x24.org/spew/m/thread-microopt-v1%40r49282.txt

target 0: 2.1.5 (ruby 2.1.5p273 (2014-11-13 revision 48405) [x86_64-linux]) at "/home/ew/ruby-2.1/bin/ruby"
target 1: trunk (ruby 2.3.0dev (2015-01-16 trunk 49282) [x86_64-linux]) at "/home/ew/rrrr/b/i/bin/ruby"
target 2: built (ruby 2.3.0dev (2015-01-16 trunk 49282) [x86_64-linux]) at "/home/ew/ruby/b/i/bin/ruby"

2.1.5 1.0573025540215895
2.1.5 1.0493981029139832
2.1.5 1.0576379200210795
trunk 1.2876477020327002
trunk 1.2424484699731693
trunk 1.2432217099703848
built 1.1531978889834136
built 1.137529328931123
built 1.1509092160267755

Elapsed time: 10.381246521 (sec)

benchmark results:
minimum results in each 3 measurements.
Execution time (sec)
name 2.1.5 trunk built
vm_thread_create_join 1.049 1.242 1.138

Speedup ratio: compare with the result of `2.1.5' (greater is better)
name trunk built
vm_thread_create_join 0.845 0.923

#4 - 01/18/2015 01:55 AM - normalperson (Eric Wong)

- Assignee set to akr (Akira Tanaka)

akr: any comments? I'll commit my patch in a few days, but I hope we
can recover more performance. Thanks.

#5 - 02/18/2015 05:31 AM - tgxworld (Guo Xiang Tan)

Eric Wong wrote:

akr: any comments? I'll commit my patch in a few days, but I hope we can recover more performance. Thanks.

Hi Eric,

Just wanted to bring your attention to bm_vm_thread_pass_flood. http://rubybench.org/ruby/ruby/commits?result_type=vm_thread_pass_flood

before_patch: 0.079s
after_patch: 0.086

The benchmark got slower after your patch so I thought you might be interested in knowing that. Not really a report but just to bring your attention to it since I don't know c well enough to actually understand how Ruby is implemented. Thank you! :)))

#6 - 02/18/2015 10:08 AM - normalperson (Eric Wong)

Thanks for the info. It seems my patch changes object allocation counts enough to throw GC off for this benchmark. Having more/less threads or other objects changes the effect.

But in general, thread scheduler benchmarks with many concurrenty threads are not very reliable in my experience (the mutex benchmarks are notoriously unreliable for me).

I think your original bm_thread_create_join is important and relevant since only one thread is running, but scheduling hundreds/thousands of threads becomes highly unpredictable with the GVL (GVL fairness improved greatly in 1.9.3).

And don't worry about not knowing C well. I only pretended to know C in the beginning. After several years, I realized I wasn't pretending :)

#7 - 03/01/2015 06:04 AM - ktsj (Kazuki Tsujimoto)

- Related to Bug #10922: TracePoint#binding may return nil in Ruby 2.2 added

#8 - 07/23/2019 03:09 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed