Ruby master - Bug #3890

ビジースレッドがあるとコンテキストスイッチが起きづらくなる

09/30/2010 09:42 AM - usa (Usaku NAKAMURA)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>ko1 (Koichi Sasada)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
<tr>
<td>Backport:</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

```ruby
def do_something
  # do something
end
```

Ruby 1.9.3dev (2010-09-29 trunk 29361) [x64-mswin64_90]

### Backport

In message "[ruby-dev:42310] ビジースレッドがあるとコンテキストスイッチが起きづらくなる" on Sep.29,2010 21:16:52, beuniv@gmail.com wrote:

```
こんにちは。
ビジー状態のスレッドがあると中々他のスレッドに処理が戻らず実行速度が遅くなることがあります。
環境はruby 1.9.3dev (2010-09-29 trunk 29361) [x64-mswin64]です。
以下のスクリプトを実行すると終了するまでに時間がかかります。
```

```ruby
  t = Time.now
  n = 0
  Thread.new{loop {n+=1} }
  Thread.new{}.join
  p [n, Time.now - t]
  
  ruby.exe a.rb
  [82741230, 10.6304]
```

追ってみたところnative_sleepのGVL_UNLOCK_ENDで、mainスレッドが
GVLを取得できずに止まっている時間が長いようです。
ためしにCriticalSectionではなく、USE_WIN32_MUTEXで、Mutexを使うように変更するとすぐ終わるようになりました。

MutexLock
ruby.exe a.rb
[65482, 0.017]

```
ruby/test_threads.rb:test_listが結構な割合で失敗するので気がつきました。
```

Kuwamoto

```
--
U.Nakamura usa@garbagecollect.jp
=end
```

### Related issues:

Related to Ruby master - Bug #2359: test_threads.rb:test_list Closed 11/12/2009

### Associated revisions
Revision 450463d5 - 11/27/2010 08:15 PM - ko1 (Koichi Sasada)

- thread.c, vm_core.h: make gvl_acquire/release/init/destruct APIs to modularize GVL implementation.
- thread_pthread.c, thread_pthread.h: Two GVL implementations. (1) Simple locking GVL which is same as existing GVL. (2) Wake-up queued threads. The wake-up order is simple FIFO. (We can make several queues to support exact priorities, however this causes some issues such as priority inversion and so on.) This impl. prevents spin-loop (*1) caused on SMP environments. *1: Only one Ruby thread acquires GVL again and again. Bug #2359 [ruby-core:26694]
- thread_win32.c, thread_win32.h: Using simple lock not by CRITICAL_SECTION but by Mutex. Bug #3890 [ruby-dev:42315]
- vm.c (ruby_vm_destruct): ditto.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@29956 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 29956 - 11/27/2010 08:15 PM - ko1 (Koichi Sasada)

- thread.c, vm_core.h: make gvl_acquire/release/init/destruct APIs to modularize GVL implementation.
- thread_pthread.c, thread_pthread.h: Two GVL implementations. (1) Simple locking GVL which is same as existing GVL. (2) Wake-up queued threads. The wake-up order is simple FIFO. (We can make several queues to support exact priorities, however this causes some issues such as priority inversion and so on.) This impl. prevents spin-loop (*1) caused on SMP environments. *1: Only one Ruby thread acquires GVL again and again. Bug #2359 [ruby-core:26694]
- thread_win32.c, thread_win32.h: Using simple lock not by CRITICAL_SECTION but by Mutex. Bug #3890 [ruby-dev:42315]
- vm.c (ruby_vm_destruct): ditto.

Revision 29956 - 11/27/2010 08:15 PM - ko1 (Koichi Sasada)

- thread.c, vm_core.h: make gvl_acquire/release/init/destruct APIs to modularize GVL implementation.
- thread_pthread.c, thread_pthread.h: Two GVL implementations. (1) Simple locking GVL which is same as existing GVL. (2) Wake-up queued threads. The wake-up order is simple FIFO. (We can make several queues to support exact priorities, however this causes some issues such as priority inversion and so on.) This impl. prevents spin-loop (*1) caused on SMP environments. *1: Only one Ruby thread acquires GVL again and again. Bug #2359 [ruby-core:26694]
- thread_win32.c, thread_win32.h: Using simple lock not by CRITICAL_SECTION but by Mutex. Bug #3890 [ruby-dev:42315]
- vm.c (ruby_vm_destruct): ditto.

Revision 29956 - 11/27/2010 08:15 PM - ko1 (Koichi Sasada)

- thread.c, vm_core.h: make gvl_acquire/release/init/destruct APIs to modularize GVL implementation.
- thread_pthread.c, thread_pthread.h: Two GVL implementations. (1) Simple locking GVL which is same as existing GVL. (2) Wake-up queued threads. The wake-up order is simple FIFO. (We can make several queues to support exact priorities, however this causes some issues such as priority inversion and so on.) This impl. prevents spin-loop (*1) caused on SMP environments. *1: Only one Ruby thread acquires GVL again and again. Bug #2359 [ruby-core:26694]
- thread_win32.c, thread_win32.h: Using simple lock not by CRITICAL_SECTION but by Mutex. Bug #3890 [ruby-dev:42315]
- vm.c (ruby_vm_destruct): ditto.

Revision 29956 - 11/27/2010 08:15 PM - ko1 (Koichi Sasada)

- thread.c, vm_core.h: make gvl_acquire/release/init/destruct APIs to modularize GVL implementation.
- thread_pthread.c, thread_pthread.h: Two GVL implementations. (1) Simple locking GVL which is same as existing GVL. (2) Wake-up queued threads. The wake-up order is simple FIFO. (We can make several queues to support exact priorities, however this causes some issues such as priority inversion and so on.) This impl. prevents spin-loop (*1) caused on SMP environments. *1: Only one Ruby thread acquires GVL again and again. Bug #2359 [ruby-core:26694]
- thread_win32.c, thread_win32.h: Using simple lock not by CRITICAL_SECTION but by Mutex. Bug #3890 [ruby-dev:42315]
- vm.c (ruby_vm_destruct): ditto.

Revision 29956 - 11/27/2010 08:15 PM - ko1 (Koichi Sasada)

- thread.c, vm_core.h: make gvl_acquire/release/init/destruct APIs to modularize GVL implementation.
- thread_pthread.c, thread_pthread.h: Two GVL implementations. (1) Simple locking GVL which is same as existing GVL. (2) Wake-up queued threads. The wake-up order is simple FIFO. (We can make several queues to support exact priorities, however this causes some issues such as priority inversion and so on.) This impl. prevents spin-loop (*1) caused on SMP environments. *1: Only one Ruby thread acquires GVL again and again. Bug #2359 [ruby-core:26694]
**thread_win32.c, thread_win32.h**: Using simple lock not by CRITICAL_SECTION but by Mutex. Bug #3890 [ruby-dev:42315]

**vm.c (ruby_vm_destruct)**: ditto.

---

**History**

#1 - 09/30/2010 10:18 AM - usa (Usaku NAKAMURA)
- Category set to core
- Status changed from Open to Assigned
- Assignee set to usa (Usaku NAKAMURA)
- Target version set to 2.0.0
- ruby -v set to ruby 1.9.3dev (2010-09-29 trunk 29361) [x64-mswin64_90]

```ruby
=begin
=end

#2 - 09/30/2010 08:49 PM - beuniv (shintaro kuwamoto)

```ruby
=begin
こんにちは。
コア数のようですね。Corei7 940 HT有りで見かけ上8コアで動かしてます。
affinity maskを設定して動かすと3コアぐらいから競合が増えているように見えます。
時間は結構ぶれがあるので参考程度です。

start /affinity 0x1 .uby.exe a.rb
start /affinity 0x3 .uby.exe a.rb
start /affinity 0x7 .uby.exe a.rb
start /affinity 0xF .uby.exe a.rb
start /affinity 0x1F .uby.exe a.rb
start /affinity 0x3F .uby.exe a.rb
start /affinity 0x7F .uby.exe a.rb
start /affinity 0xFF .uby.exe a.rb
```

<table>
<thead>
<tr>
<th>CPU COUNT</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[491032, 0.11]</td>
</tr>
<tr>
<td>2</td>
<td>[92323, 0.013]</td>
</tr>
<tr>
<td>3</td>
<td>[6901444, 0.8736]</td>
</tr>
<tr>
<td>4</td>
<td>[8894934, 1.56]</td>
</tr>
<tr>
<td>5</td>
<td>[6975323, 0.9984]</td>
</tr>
<tr>
<td>6</td>
<td>[124533026, 15.7444]</td>
</tr>
<tr>
<td>7</td>
<td>[48895362, 5.981]</td>
</tr>
<tr>
<td>8</td>
<td>[9636525, 1.2842]</td>
</tr>
</tbody>
</table>

```ruby
=begin
=end

#3 - 10/01/2010 01:57 AM - ko1 (Koichi Sasada)

```ruby
=begin
(2010/09/30 12:48), kuwamoto shintaro wrote:

```ruby
=begin
=end

09/13/2021
もう少ししたら、実装を直そうと思いますので、その時に、またお試し頂ければと思います。

---

// SASADA Koichi at atdot dot net
=end

#4 - 10/01/2010 10:02 AM - usa (Usaku NAKAMURA)
- Assignee changed from usa (Usaku NAKAMURA) to ko1 (Koichi Sasada)

=begin

---

// SASADA Koichi at atdot dot net
=end

#5 - 10/01/2010 01:41 PM - beuniv (shintaro kuwamoto)

=begin

この問題に関しては、まさに native_thread_yield() の実装というか、その辺の実装(GVLの受け渡しの実装)の問題として、既知のものです。

なるほど了解です。

ついでですが、

http://redmine.ruby-lang.org/issues/show/2359

---

// SASADA Koichi at atdot dot net
=end

#6 - 11/28/2010 05:16 AM - ko1 (Koichi Sasada)

=begin

(2010/09/30 9:42), U.Nakamura wrote:

ためしにCriticalSectionではなく、USE_WIN32_MUTEXで、Mutexを使うように変更するとすぐ終わるようになりました。

安易ですが、mutex を使うようにしてみました。どうでしょうか。

---

// SASADA Koichi at atdot dot net
=end

#7 - 11/28/2010 05:17 AM - ko1 (Koichi Sasada)

- Status changed from Assigned to Closed
- % Done changed from 0 to 100

=begin

This issue was solved with changeset r29956.
Usaku, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

=end

#8 - 11/29/2010 01:43 PM - beuniv (shintaro kuwamoto)

=begin

ruby 1.9.3dev (2010-11-28 trunk 29965) [x86-mswin64_90]

---

// SASADA Koichi at atdot dot net
=end