monitor.rb introduces MonitorMixin module and Monitor class. Now, rubygems.rb requires monitor.rb so that most of Ruby applications (*1) requires monitor.rb

[*1] running MRI with --disable-gems can eliminate monitor.rb, but only a few people use this option.

Also recent changes (6ec1720aa38772ba4b7515790443cb578340518b and so on) to make it atomic introduces several overhead. Making them in C will avoid such overhead.

Naruse-san made monitor-ext gem to make it faster by making C-extension. This proposal is to make it built-in class.

Approach

Implement Monitor class as re-entrant mutex object in C and implement MonitorMixin with it. Now, Monitor class is implemented with MonitorMixin, but it is easy to implement Monitor class directly and faster in C.

MonitorMixin and MonitorMixin::ConditionalVariable are remained in lib/monitor.rb, because it is easy to write :p

Implementation

https://github.com/ruby/ruby/compare/master...ko1:thread_monitor

Evaluation

require_relative 'benchmark/lib/load'
Benchmark.driver(repeat_count: 1){|x|  
  x.executable name: '2.5.2', command: `%w'/home/ko1/ruby/install/ruby_2_5/bin/ruby'  
  x.executable name: '2.6.6', command: `%w'/home/ko1/ruby/install/ruby_2_6/bin/ruby'  
  x.executable name: 'trunk', command: `%w'/home/ko1/ruby/install/trunk/bin/ruby'

  x.prelude %{  
    mon = Monitor.new  
  }

  x.report %{  
    mon.synchronize{  
      mon.synchronize{  
      }  
    }  
  }  

  #=>  
  # trunk:  4345545.8 i/s  
  # 2.5.2:  1657762.1 i/s - 2.62x slower  
  # 2.6.6:  499165.7 i/s - 8.71x slower
Queue, Mutex are the alias of Thread::Queue, Thread::Mutex.

p Queue #=> Thread::Queue

I defined Thread::Monitor and make alias by Monitor = Thread::Monitor. Is it okay?

I removed mon_prefix methods from Monitor class. I believe they are only for MonitorMixin.

... but I found two examples which call mon_prefix methods for Monitor class:

```
/home/gem-codesearch/gem-codesearch/latest-gem/gel-0.3.0/lib/gel/db.rb:   @monitor.mon_owned?
/home/gem-codesearch/gem-codesearch/latest-gem/keepyourhead-0.2.2/lib/Keepyourhead/Cache.rb:
   @monitor.mon_synchronize { ...
```

Should we obsolete them for Monitor class?

Associated revisions

Revision caac5f77 - 10/19/2019 07:52 PM - ko1 (Koichi Sasada)
make monitor.so for performance. (#2576)
Recent monitor.rb has performance problem because of interrupt handlers. 'Monitor#synchronize' is frequently used primitive so the performance of this method is important.
This patch rewrite 'monitor.rb' with 'monitor.so' (C-extension) and make it faster. See [Feature #16255] for details.
Monitor class objects are normal object which include MonitorMixin.
This patch introduce a Monitor class which is implemented on C and MonitorMixin uses Monitor object as re-entrant (recursive) Mutex. This technique improve performance because we don't need to care atomicity and we don't need accesses to instance variables any more on Monitor class.

History

#1 - 10/17/2019 06:53 PM - ko1 (Koichi Sasada)
Matz wants to make it as an extension library, so I moved it as extension library.

#2 - 10/19/2019 07:52 PM - ko1 (Koichi Sasada)
- Status changed from Open to Closed

Applied in changeset git|caac5f77ae288b5982708b8690e712e1cae0cf6.

make monitor.so for performance. (#2576)
Recent monitor.rb has performance problem because of interrupt handlers. 'Monitor#synchronize' is frequently used primitive so the performance of this method is important.
This patch rewrite 'monitor.rb' with 'monitor.so' (C-extension) and make it faster. See [Feature #16255] for details.
Monitor class objects are normal object which include MonitorMixin.
This patch introduce a Monitor class which is implemented on C and MonitorMixin uses Monitor object as re-entrant (recursive) Mutex. This technique improve performance because we don't need to care atomicity and we don't need accesses to instance variables any more on Monitor class.

#3 - 12/01/2019 05:20 AM - y-yagi (Yuji Yaginuma)
This change breaks the library that extends Monitor class. For example, Rails has such class and it breaks in Ruby 2.7. [https://github.com/rails/rails/blob/4d1aae05ff1f344f775e6367b9a9add4dda37746b5c/accesstors/lib/active_support/concurrency/load_interlock_aware_monitor.rb#L7-L15](https://github.com/rails/rails/blob/4d1aae05ff1f344f775e6367b9a9add4dda37746b5c/accesstors/lib/active_support/concurrency/load_interlock_aware_monitor.rb#L7-L15)

Isn't it guaranteed to make such extending to Monitor?
(I'm not sure if Monitor class is a public API or not)
The latest google-cloud-pubsub breaks too because new_cond comes before initialization:
https://github.com/googleapis/google-cloud-ruby/blob/google-cloud-pubsub/v1.1.2/google-cloud-pubsub/lib/google/cloud/pubsub/async_publisher.rb#L92-L95

Could you share the error and backtrace in both of these cases?

Rails encountered the same issue (related to https://github.com/rails/rails/pull/37822).
The error and backtrace of in this case are as follows.
https://buildkite.com/rails/rails/builds/65193#02e136eb-edea-4367-aee0-77e0a82f8531/990-1200

In the first case, the method did not call as expected, so it does not show error and backtrace related to the issue.

I fixed new_cond before mon_initialize problem c6e3db0c66312af1e932c21006437419efa9ac75
https://github.com/rails/rails/blob/461aae05f194f775e63e7be9aad4dda37746bd5c/activesupport/lib/active_support/concurrency/load_interlock_aban
This code depends "how to implement #synchronize" so that I'm not sure MRI should support it for compatibility (I don't think so, if most of people
doesn't use this technique).

Thanks for your response! I understood the support policy.