Ruby master - Misc #16160

Lazy init thread local storage

09/09/2019 09:36 PM - methodmissing (Lourens Naudé)

Status:	Open	
Priority:	Normal	
Assignee:		

Description

References PR https://github.com/ruby/ruby/pull/2295

Why?

The local_storage member of execution context is lazy initialized and drives the Thread#[] and Thread#[]= APIs, which are Fiber local and not Thread local storage. I think the same lazy init pattern should be applied to the APIs below as well - reduces one Hash alloc per thread created that does not use thread locals.

Lazy allocates thread local storage for the following APIs

- Thread#thread_variable_get - early returns nil on locals Hash not initialised
- Thread#thread_variable_set - forces allocation of the locals Hash if not initilalised
- Thread#thread_variables - early returns the empty array AND saves on Hash iteration if locals Hash not initialised
- Thread#thread_variable? - early returns false on locals Hash not initialised

Other notes

- Moved initial implementation from internal.h to thread.c local to call sites.
- Preferred defs/id.def for the locals ID (seeing this pattern used more often, but not sure if that is preferred to inline rb_intern yet. Either way there's quite a few different conventions around IDs in the codebase at the moment and happy to help converging to a standard instead.
- Maybe a flag is overkill and NIL_P on locals ivar could also work ...

Thoughts?

History

#1 - 09/19/2019 07:52 AM - nobu (Nobuyoshi Nakada)

I'm positive about this, except for the performance.
Do you have any numbers?

#2 - 09/22/2019 01:55 AM - methodmissing (Lourens Naudé)

nobu (Nobuyoshi Nakada) wrote:

> I'm positive about this, except for the performance.
> Do you have any numbers?

Apologies for the delay in replying.

Using benchmark-driver script (running set last as it would taint the others by initializing the locals table on the thread object initialized in prelude):

```
prelude: |
  th = Thread.new {}
benchmark:
  thread_variable_get: th.thread_variable_get('foo')
  thread_variables: th.thread_variables
  thread_variable_p: th.thread_variable?('foo')
  thread_variable_set: th.thread_variable_set('foo', 'bar')
loop_count: 1000000
```

```
lourens@CarbonX1:~/src/ruby/ruby$ /usr/local/bin/ruby --disable=gems -rrubygems -I./benchmark/lib ./benchmark/
benchmark-driver/exe/benchmark-driver             --executables="compare-ruby::~/src/ruby/trunk/ruby --disable
=gems -I.ext/common --disable-gem"            --executables="built-ruby::./miniruby -I./lib -I. -I.ext/common
  -r./prelude --disable-gem" -v --repeat-count=10 $HOME/src/lazy_init_thread_locals.yml
compare-ruby: ruby 2.7.0dev (2019-09-22T01:11:51Z master a0ce0b6297) [x86_64-linux]
```

```
built-ruby: ruby 2.7.0dev (2019-09-22T01:21:06Z lazy-init-thread-l.. 24463b7252) [x86_64-linux]
Calculating -------------------------------------
                     compare-ruby  built-ruby
 thread_variable_get      11.305M     33.901M i/s -      1.000M times in 0.088456s 0.029498s
    thread_variables      22.765M     40.344M i/s -      1.000M times in 0.043927s 0.024787s
   thread_variable_p      19.260M     20.883M i/s -      1.000M times in 0.051921s 0.047886s
 thread_variable_set       8.195M      8.543M i/s -      1.000M times in 0.122030s 0.117054s

Comparison:
              thread_variable_get
          built-ruby:  33900806.8 i/s
        compare-ruby:  11305022.7 i/s - 3.00x  slower

                 thread_variables
          built-ruby:  40344251.1 i/s
        compare-ruby:  22765106.8 i/s - 1.77x  slower

                thread_variable_p
          built-ruby:  20882884.5 i/s
        compare-ruby:  19260142.8 i/s - 1.08x  slower

              thread_variable_set
          built-ruby:   8543090.8 i/s
        compare-ruby:   8194725.4 i/s - 1.04x  slower
```

A regression on thread_variable_set, but improvement on others.

And with memory runner (although knowing ahead of time it's just the hash, 40 bytes with array table saved):

```
lourens@CarbonX1:~/src/ruby/ruby$ /usr/local/bin/ruby --disable=gems -rrubygems -I./benchmark/lib ./benchmark/
benchmark-driver/exe/benchmark-driver             --executables="compare-ruby::~/src/ruby/trunk/ruby --disable
=gems -I.ext/common --disable-gem"             --executables="built-ruby::./miniruby -I./lib -I. -I.ext/common
  -r./prelude --disable-gem" -v --repeat-count=10 -r memory $HOME/src/lazy_init_thread_locals.yml
compare-ruby: ruby 2.7.0dev (2019-09-22T01:11:51Z master a0ce0b6297) [x86_64-linux]
built-ruby: ruby 2.7.0dev (2019-09-22T01:21:06Z lazy-init-thread-l.. 24463b7252) [x86_64-linux]
Calculating -------------------------------------
                     compare-ruby  built-ruby
 thread_variable_get      11.632M     11.528M bytes -      1.000M times
    thread_variables      11.668M     11.472M bytes -      1.000M times
   thread_variable_p      11.692M     11.452M bytes -      1.000M times
 thread_variable_set      11.652M     11.568M bytes -      1.000M times

Comparison:
              thread_variable_get
          built-ruby:  11528000.0 bytes
        compare-ruby:  11632000.0 bytes - 1.01x  larger

                 thread_variables
          built-ruby:  11472000.0 bytes
        compare-ruby:  11668000.0 bytes - 1.02x  larger

                thread_variable_p
          built-ruby:  11452000.0 bytes
        compare-ruby:  11692000.0 bytes - 1.02x  larger

              thread_variable_set
          built-ruby:  11568000.0 bytes
        compare-ruby:  11652000.0 bytes - 1.01x  larger
```