Since r62466 ("thread_pthread.c: shorten and fix thread cache implementation"), our thread cache is no longer buggy with programs using fork. This makes significant improvements in vm_thread_alive_check1 and vm_thread_create_join benchmarks and does not introduce regressions.

Unlike old thread cache, I've changed the cache to only last 3 seconds since per-thread setup in most programs rarely takes more than a few milliseconds to re-establish things like network connections. This is configurable by changing the THREAD_CACHE_TIME variable.

I hope this allows users to simplify their code by removing the need for thread pools in many cases.

This seems pretty trivial and low-risk, will commit in a few days if no objections.
thread_pthread.c: enable thread cache by default

Since r62466 ("thread_pthread.c: shorten and fix thread cache implementation"), our thread cache is no longer buggy with programs using fork. This makes significant improvements in vm_thread_alive_check1 and vm_thread_create_join benchmarks and does not introduce regressions.

Unlike old thread cache, I've changed the cache to only last 3 seconds since per-thread setup in most programs rarely takes more than a few milliseconds to re-establish things like network connections. This is configurable by changing the THREAD_CACHE_TIME variable.

I hope this allows users to simplify their code by removing the need for thread pools in many cases.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Built</th>
<th>Trunk</th>
</tr>
</thead>
<tbody>
<tr>
<td>vm_thread_alive_check1</td>
<td>10.872</td>
<td>0.150</td>
</tr>
<tr>
<td>vm_thread_close</td>
<td>1.988</td>
<td>2.027</td>
</tr>
<tr>
<td>vm_thread_condvar1</td>
<td>0.751</td>
<td>0.767</td>
</tr>
<tr>
<td>vm_thread_condvar2</td>
<td>0.744</td>
<td>0.752</td>
</tr>
<tr>
<td>vm_thread_create_join</td>
<td>5.296</td>
<td>2.343</td>
</tr>
<tr>
<td>vm_thread_mutex1</td>
<td>1.911</td>
<td>1.892</td>
</tr>
<tr>
<td>vm_thread_mutex2</td>
<td>1.902</td>
<td>1.896</td>
</tr>
<tr>
<td>vm_thread_mutex3</td>
<td>2.389</td>
<td>2.313</td>
</tr>
<tr>
<td>vm_thread_pass</td>
<td>0.271</td>
<td>0.272</td>
</tr>
<tr>
<td>vm_thread_pass_flood</td>
<td>0.175</td>
<td>0.179</td>
</tr>
<tr>
<td>vm_thread_pipe</td>
<td>0.460</td>
<td>0.436</td>
</tr>
<tr>
<td>vm_thread_queue</td>
<td>0.453</td>
<td>0.446</td>
</tr>
<tr>
<td>vm_thread_sized_queue</td>
<td>0.547</td>
<td>0.547</td>
</tr>
<tr>
<td>vm_thread_sized_queue2</td>
<td>1.417</td>
<td>1.413</td>
</tr>
<tr>
<td>vm_thread_sized_queue3</td>
<td>1.410</td>
<td>1.426</td>
</tr>
<tr>
<td>vm_thread_sized_queue4</td>
<td>0.787</td>
<td>0.791</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Built</th>
<th>Trunk</th>
</tr>
</thead>
<tbody>
<tr>
<td>vm_thread_alive_check1</td>
<td>72.456</td>
<td></td>
</tr>
<tr>
<td>vm_thread_close</td>
<td>0.981</td>
<td></td>
</tr>
<tr>
<td>vm_thread_condvar1</td>
<td>0.979</td>
<td></td>
</tr>
<tr>
<td>vm_thread_condvar2</td>
<td>0.990</td>
<td></td>
</tr>
<tr>
<td>vm_thread_create_join</td>
<td>2.260</td>
<td></td>
</tr>
<tr>
<td>vm_thread_mutex1</td>
<td>1.010</td>
<td></td>
</tr>
<tr>
<td>vm_thread_mutex2</td>
<td>1.003</td>
<td></td>
</tr>
<tr>
<td>vm_thread_mutex3</td>
<td>1.033</td>
<td></td>
</tr>
<tr>
<td>vm_thread_pass</td>
<td>0.994</td>
<td></td>
</tr>
<tr>
<td>vm_thread_pass_flood</td>
<td>0.980</td>
<td></td>
</tr>
<tr>
<td>vm_thread_pipe</td>
<td>1.055</td>
<td></td>
</tr>
<tr>
<td>vm_thread_queue</td>
<td>1.016</td>
<td></td>
</tr>
<tr>
<td>vm_thread_sized_queue</td>
<td>0.999</td>
<td></td>
</tr>
<tr>
<td>vm_thread_sized_queue2</td>
<td>1.003</td>
<td></td>
</tr>
<tr>
<td>vm_thread_sized_queue3</td>
<td>0.989</td>
<td></td>
</tr>
<tr>
<td>vm_thread_sized_queue4</td>
<td>0.995</td>
<td></td>
</tr>
</tbody>
</table>

[ruby-core:87030] [Feature #14757]
variable.

I hope this allows users to simplify their code by removing the need for thread pools in many cases.

```plaintext
vm_thread_alive_check1  10.872   0.150
vm_thread_close         1.988   2.027
vm_thread_condvar1      0.751   0.767
vm_thread_condvar2      0.744   0.752
vm_thread_create_join   5.296   2.343
vm_thread_mutex1        1.911   1.892
vm_thread_mutex2        1.902   1.896
vm_thread_mutex3        2.389   2.313
vm_thread_pass          0.271   0.272
vm_thread_pass_flood    0.175   0.179
vm_thread_queue         0.460   0.436
vm_thread_sized_queue   0.547   0.547
```

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

```
name    built
vm_thread_alive_check1 72.456
vm_thread_close         0.981
vm_thread_condvar1      0.979
vm_thread_condvar2      0.990
vm_thread_create_join   2.260
vm_thread_mutex1        1.010
vm_thread_mutex2        1.003
vm_thread_mutex3        1.033
vm_thread_pass          0.994
vm_thread_pass_flood    0.980
vm_thread_queue         1.055
```

Revision 63498 - 05/23/2018 09:16 PM - normal

```
thread_pthread.c: enable thread cache by default

Since r62466 ("thread_pthread.c: shorten and fix thread cache implementation"),
our thread cache is no longer buggy with programs using fork.
This makes significant improvements in vm_thread_alive_check1
and vm_thread_create_join benchmarks and does not introduce
regressions.

Unlike old thread cache, I've changed the cache to only last 3
seconds since per-thread setup in most programs rarely takes
more than a few milliseconds to re-establish things like network
connections. This is configurable by changing the THREAD_CACHE_TIME
variable.

I hope this allows users to simplify their code by removing the
need for thread pools in many cases.
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Trunk</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>vm_thread_alive_check1</td>
<td>72.456</td>
<td></td>
</tr>
<tr>
<td>vm_thread_close</td>
<td>0.981</td>
<td></td>
</tr>
<tr>
<td>vm_thread_condvar1</td>
<td>0.979</td>
<td></td>
</tr>
<tr>
<td>vm_thread_condvar2</td>
<td>0.990</td>
<td></td>
</tr>
<tr>
<td>vm_thread_create_join</td>
<td>2.260</td>
<td></td>
</tr>
<tr>
<td>vm_thread_mutex1</td>
<td>1.010</td>
<td></td>
</tr>
<tr>
<td>vm_thread_mutex2</td>
<td>1.003</td>
<td></td>
</tr>
<tr>
<td>vm_thread_mutex3</td>
<td>1.033</td>
<td></td>
</tr>
<tr>
<td>vm_thread_pass</td>
<td>0.994</td>
<td></td>
</tr>
<tr>
<td>vm_thread_pass_flood</td>
<td>0.980</td>
<td></td>
</tr>
<tr>
<td>vm_thread_pipe</td>
<td>1.055</td>
<td></td>
</tr>
<tr>
<td>vm_thread_queue</td>
<td>1.016</td>
<td></td>
</tr>
<tr>
<td>vm_thread_sized_queue</td>
<td>0.999</td>
<td></td>
</tr>
<tr>
<td>vm_thread_sized_queue2</td>
<td>1.003</td>
<td></td>
</tr>
<tr>
<td>vm_thread_sized_queue3</td>
<td>0.989</td>
<td></td>
</tr>
<tr>
<td>vm_thread_sized_queue4</td>
<td>0.995</td>
<td></td>
</tr>
</tbody>
</table>

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

**Revision 9e5354eb - 05/29/2018 01:18 AM - normal**

NEWS: add item about built-in thread cache

This change may impact design of future programs and obviate thread pools in many cases, so it's worth a mention. Anyways, this seems stable since r63499 [Feature #14757]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63516 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

**Revision 63516 - 05/29/2018 01:18 AM - normalperson (Eric Wong)**

NEWS: add item about built-in thread cache

This change may impact design of future programs and obviate thread pools in many cases, so it's worth a mention. Anyways, this seems stable since r63499 [Feature #14757]

**Revision 63516 - 05/29/2018 01:18 AM - normal**

NEWS: add item about built-in thread cache

This change may impact design of future programs and obviate thread pools in many cases, so it's worth a mention. Anyways, this seems stable since r63499 [Feature #14757]

**Revision c5cb386e - 05/30/2018 11:49 AM - normal**

NEWS: clarify item about built-in thread cache

[Feature #14757]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63527 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

**Revision 63527 - 05/30/2018 11:49 AM - normalperson (Eric Wong)**

NEWS: clarify item about built-in thread cache

[Feature #14757]

**Revision 63527 - 05/30/2018 11:49 AM - normal**

NEWS: clarify item about built-in thread cache

[Feature #14757]

**History**

#1 - 05/15/2018 07:12 AM - normalperson (Eric Wong)

Oops, I forgot to squash the patch and only sent the second part :x
Anyways, it is trivial:

```c
--- a/thread_pthread.c
+++ b/thread_pthread.c
@@ -432,7 +432,7 @@ native_thread_destroy(rb_thread_t *th)
 }

 #ifndef USE_THREAD_CACHE
-#define USE_THREAD_CACHE 0
+#define USE_THREAD_CACHE 1
 #endif

 #if USE_THREAD_CACHE
```

#2 - 05/23/2018 09:17 PM - normalperson (Eric Wong)

- Status changed from Open to Closed

Applied in changeset trunk\r63498.

thread_pthread.c: enable thread cache by default

Since r62466 ("thread_pthread.c: shorten and fix thread cache implementation"), our thread cache is no longer buggy with programs using fork. This makes significant improvements in vm_thread_alive_check1 and vm_thread_create_join benchmarks and does not introduce regressions.

Unlike old thread cache, I've changed the cache to only last 3 seconds since per-thread setup in most programs rarely takes more than a few milliseconds to re-establish things like network connections. This is configurable by changing the THREAD_CACHE_TIME variable.

I hope this allows users to simplify their code by removing the need for thread pools in many cases.

```text
vm_thread_alive_check1 10.872   0.150
vm_thread_close         1.988   2.027
vm_thread_condvar1     0.751   0.767
vm_thread_condvar2     0.744   0.752
vm_thread_create_join  5.296   2.343
vm_thread_mutex1       1.911   1.892
vm_thread_mutex2       1.902   1.896
vm_thread_mutex3       2.389   2.313
vm_thread_pass         0.271   0.272
vm_thread_pass_flood   0.175   0.179
vm_thread_pipe         0.460   0.436
vm_thread_queue        0.453   0.446
vm_thread_sized_queue  0.547   0.547
vm_thread_sized_queue2 1.417   1.413
vm_thread_sized_queue3 1.410   1.426
vm_thread_sized_queue4 0.787   0.791
```

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

```text
name    built    name    built
vm_thread_alive_check1 72.456  vm_thread_close  0.981
vm_thread_close         1.000  vm_thread_condvar1  0.979
vm_thread_condvar2     0.990   vm_thread_create_join 2.260
vm_thread_mutex1       1.010   vm_thread_mutex1   1.010
vm_thread_mutex2       1.003   vm_thread_mutex2   1.003
vm_thread_mutex3       1.033   vm_thread_mutex3   1.033
vm_thread_pass         0.994   vm_thread_pass    0.980
vm_thread_pass_flood   0.980   vm_thread_pass_flood 1.055
vm_thread_queue        1.016   vm_thread_queue    1.016
vm_thread_sized_queue  0.999   vm_thread_sized_queue 0.999
vm_thread_sized_queue2 1.003   vm_thread_sized_queue2 1.003
vm_thread_sized_queue3 0.989   vm_thread_sized_queue3 0.989
vm_thread_sized_queue4 0.995   vm_thread_sized_queue4 0.995
```

[ruby-core:87030] [Feature #14757]
ko1 brought up a good point: this may interact badly with 3rd-party libraries which use thread-local storage via pthread_getspecific/pthread_setspecific (or compiler extensions).

On one hand, I expected this to be beneficial for malloc implementations which use thread-specific storage (most all of them do); but there may be problems in 3rd-party libraries we do not know about...

So, we might need to disable this before 2.6 release :<

ko1 and I agreed this is a low-risk and will keep thread cache enabled for 2.6. pthread_(get,set)specific isn't very popular in 3rd-party libraries, or they use it in ways which does not conflict with thread-caching.