Thread#thread_id to get associated native thread id (LWP.) It might return nil if OS doesn't support thread id or equivalent.

When I tried to investigate which Ruby thread of an application is busy, I did the following steps:

- checked the CPU usage of the Ruby application's threads using `ps -eLf` or top (with H key) and got which thread is busy
- dumped all the threads of the application using `https://github.com/frsyuki/sigdump`
- tried to find a busy thread in the thread dump result, but the thread dump doesn't contain thread id...

Thread class itself has no method to get associated thread id. If the class has #thread_id or something, I can create a PR on sigdump project to add thread id in thread dump output to make investigations with thread dump much easier.

Thread#name may seem an alternative to Thread#thread_id, but Thread#name just returns a value that's set through Thread#name= while Thread#thread_id returns a thread id that OS or something has assigned.

In case of Java, thread dump created by `jcmd ${pid} Thread.print` contains nid field which is an associated native thread id:

```
"http-bio-80-exec-77" daemon prio=6 tid=0x0000000026f29000 nid=0xbd0 runnable [0x0000000020c7f000]
  java.lang.Thread.State: RUNNABLE           
at java.net.SocketInputStream.socketRead0(Native Method)
  at java.net.SocketInputStream.read(Unknown Source)
  at java.net.SocketInputStream.read(Unknown Source)
  at org.apache.coyote.http11.InternalInputBuffer.fill(InternalInputBuffer.java:516)
  at org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:315)
    locked <0x00000007b16e3e88> (a org.apache.tomcat.util.net.SocketWrapper)
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:617)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:565)
```

Related issues:
- Related to Ruby master - Feature #11251: Thread#name and Thread#name=

Associated revisions
- Revision 46655156 - 05/26/2021 06:14 AM - naruse (Yui NARUSE)
  Add Thread#native_thread_id [Feature #17853]
- Revision 9ddc7674 - 05/26/2021 06:18 AM - naruse (Yui NARUSE)
  Add NEWS about 46655156dcc37509dcb69fcd0717c110eb1e624a
  - Add Thread#native_thread_id [Feature #17853]
I investigated Python for reference. Current CPython supports `threading.get_native_id` ([https://docs.python.org/3/library/threading.html#threading.get_native_id](https://docs.python.org/3/library/threading.html#threading.get_native_id)) which returns native Thread ID.

Implementation details are:

- [https://github.com/python/cpython/blob/bb3e0c240bc60fe08d332ff5955d54197f79751c/Python/thread_pthread.h#L332-L356](https://github.com/python/cpython/blob/bb3e0c240bc60fe08d332ff5955d54197f79751c/Python/thread_pthread.h#L332-L356)
- [https://github.com/python/cpython/blob/bb3e0c240bc60fe08d332ff5955d54197f79751c/Python/thread_nt.h#L241-L252](https://github.com/python/cpython/blob/bb3e0c240bc60fe08d332ff5955d54197f79751c/Python/thread_nt.h#L241-L252)

This method can be used as below:

```python
import threading
import time
import os

def p():
    while True:
        print(threading.get_native_id())
        time.sleep(1)
    print(os.getpid())

thread1 = threading.Thread(target=p)
thread2 = threading.Thread(target=p)
thread1.start()
thread2.start()
thread1.join()
thread2.join()
```

```bash
# python thread.py
15 # <- pid
16 # <- Thread id
17 # <- Thread id
```

```bash
# ps -efL
```

This would be useful for Datadog's `ddtrace` gem as well: currently we need to do quite a bit of hackery and monkey-patching to get the per-thread cpu clock information, and if we had the native id, we could use that instead to call `Process.clock_gettime`.

Two concerns.

1. `name` should include native, os or something to represent it is platform value. Maybe `native_id` same as Python's API is enough. Ruby can set internal ID for threads.
2. In future the thread model can be changed (1:1 -> M:N). In this case, `native_id` should be changed (or nil if the ruby thread is not running on native thread).

`native_id` sounds better to me. I agree with it.

As for m:n thread model, in my opinion, either volatile value or nil (this can happen when it's waiting an event in select/epoll/kqueue without assigning any thread?) sounds good to me since what I want to do is just mapping the result of Ruby Thread information to native thread resource.
current_native_id might be better since it sounds like the attribute/property can be easily changed.

#8 - 05/24/2021 10:55 AM - naruse (Yui NARUSE)
Created a proposed implementation:
https://github.com/ruby/ruby/compare/master...nurse:native_thread_id

#9 - 05/26/2021 06:15 AM - naruse (Yui NARUSE)
- Status changed from Open to Closed

Applied in changeset git|46655156dccc37509dcb69fcd0717c110eb1c624a.

Add Thread#native_thread_id [Feature #17853]

#10 - 05/27/2021 11:03 AM - ivoanjo (Ivo Anjo)
Thanks a lot for building this, naruse (Yui NARUSE)!
If I can ask for a tiny addition, it would be really cool to get the native_thread_id, if available, in Thread#inspect :)

#11 - 06/05/2021 09:29 AM - nobu (Nobuyoshi Nakada)
diff --git i/thread.c w/thread.c
index cbeff44a9d4a..c627e5c8f2a 100644
--- i/thread.c
+++ w/thread.c
@@ -3456,6 +3456,14 @@ rb_thread_to_s(VALUE thread)
     if (!NIL_P(target_th->name)) {
         rb_str_catf(str, "@"PRIsVALUE, target_th->name);
     }
+  +#if USE_NATIVE_THREAD_NATIVE_THREAD_ID
+  +  if (!rb_threadptr_dead(target_th)) {
+  +      VALUE native_id = native_thread_native_thread_id(target_th);
+  +      if (!NIL_P(native_id)) {
+  +          rb_str_catf(str, "("PRIsVALUE")", native_id);
+  +      }
+  +  }
  +  +  if ((loc = threadptr_invoke_proc_location(target_th)) != Qnil) {
  +      rb_str_catf(str, "%PRIsVALUE":%PRIsVALUE,
  +                  RARRAY_AREF(loc, 0), RARRAY_AREF(loc, 1));

#12 - 06/05/2021 11:33 AM - ivoanjo (Ivo Anjo)
nobu (Nobuyoshi Nakada) that looks great!

#13 - 06/07/2021 10:09 AM - komamitsu (Mitsunori Komatsu)
Thank you, naruse (Yui NARUSE) and nobu (Nobuyoshi Nakada)!