Replace copy coroutine with pthread implementation.

06/30/2021 04:29 AM - ioquatix (Samuel Williams)

Status: Closed
Priority: Normal
Assignee: ioquatix (Samuel Williams)
Target version: 

Description

The copy coroutine is unique in that it uses a private stack area. It is also tricky to implement as it uses setjmp/longjmp to update the stack & instruction pointer. Because of this, pointers to stack allocated data are only valid while the fiber is running, and this makes implementation tricky for any code which uses stack allocated structs.

Previously we introduced some macros for dealing with this - copy coroutines fall back to malloc/free rather than using stack locals, but the chance if this causing problems is very high. In addition, rb_protect prevents us from switching fibers, which also prevents the fiber scheduler from operating within this context.

We propose to remove the copy coroutine implementation and replace it with a pthread-based cooperative scheduled implementation. This implementation uses one native thread per fiber, but only allows one of them to execute at a time using a mutex. This approach should be more consistent with the other native implementations and allow us to simplify a number of implementation details.

History

#1 - 06/30/2021 04:29 AM - ioquatix (Samuel Williams)
This is currently blocking https://bugs.ruby-lang.org/issues/18005. Pull request: https://github.com/ruby/ruby/pull/4606.

#2 - 06/30/2021 06:36 AM - xtkoba (Tee KOBAYASHI)
+1

#3 - 07/01/2021 03:41 AM - ioquatix (Samuel Williams)
- Status changed from Open to Closed

Merged.