# "Ractor" reconsidered' reconsidered

Koichi Sasada
Cookpad Inc.

# About Koichi Sasada

- Ruby interpreter developer employed by Cookpad Inc. (2017~) with @mame
  - YARV (Ruby 1.9~)
  - Generational/Incremental GC (Ruby 2.1~)
  - Ractor (Ruby 3.0~)
  - debug.gem (Ruby 3.1~)
  - …
- Ruby Association Director (2012~)

# Our goal

- Easy and lightweight parallel and concurrent programming on Ruby
  - Ractor system
  - M:N threads implementation

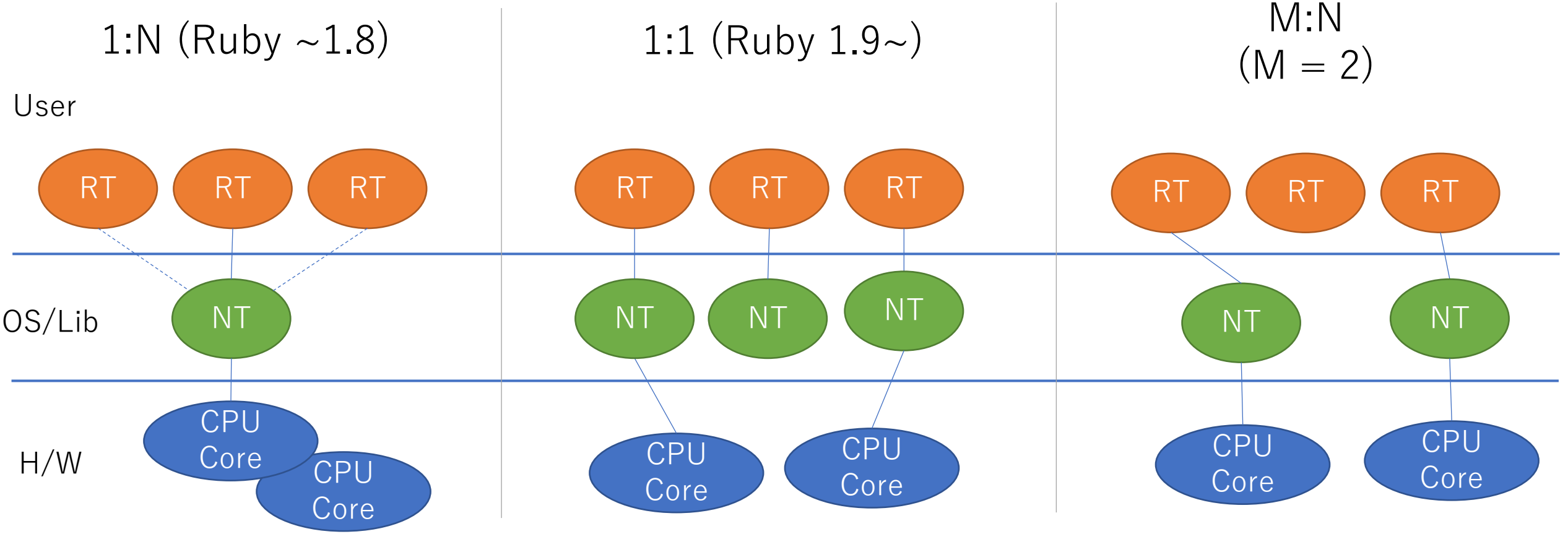
Quoted from RubyKaigi 2022 slides (modified)

Study in computer science/OS area

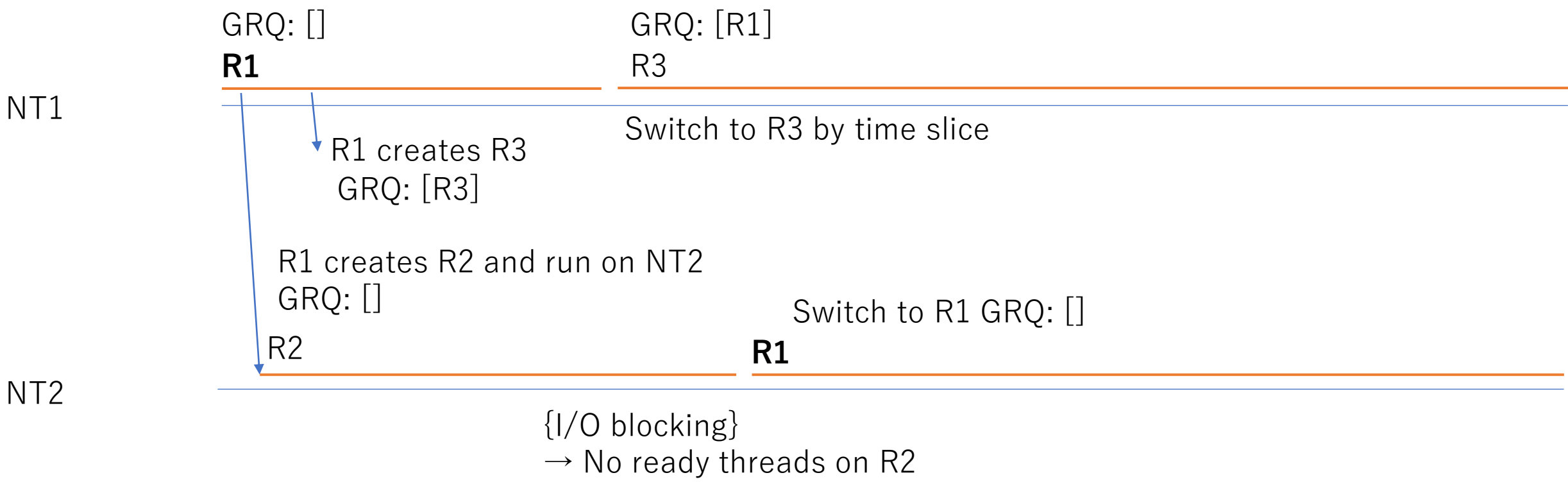
NT: Native thread or kernel thread

# MaNy project
# Thread system implementation techniques

How to handle N=3 Ruby threads/ractors (RTs) on 2 CPU cores?



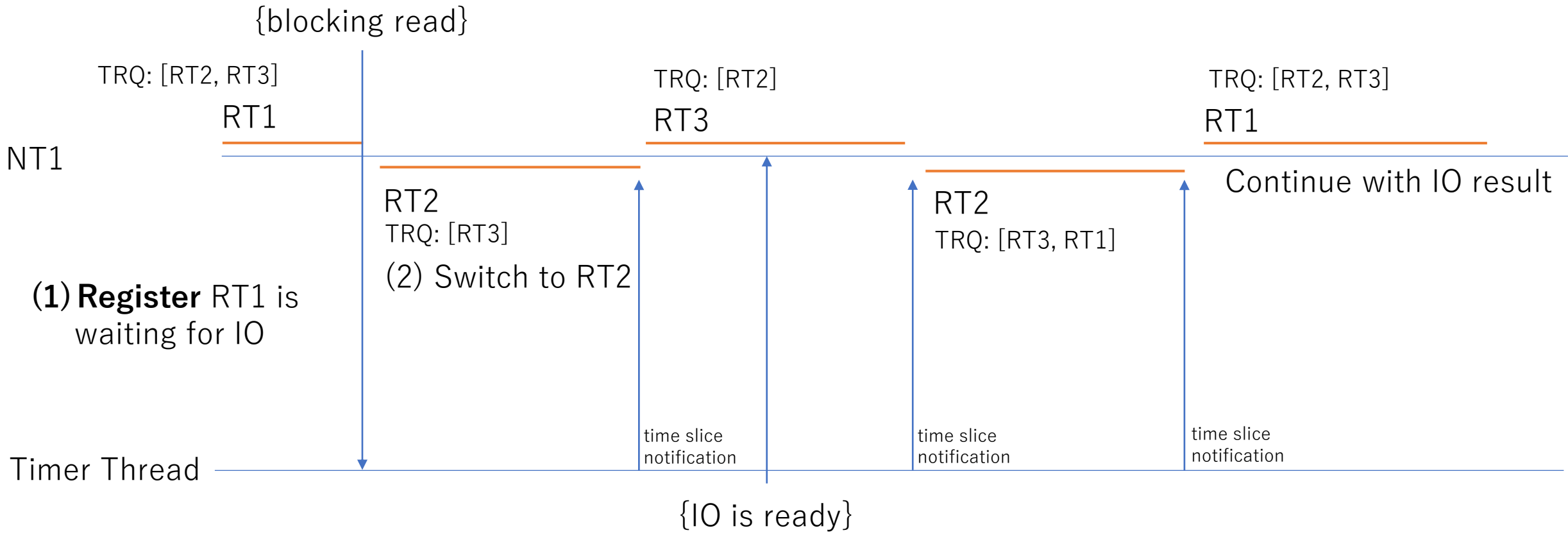
# M:N Ractor level scheduling (M=2)




📌 Ractor R1, R2, R3 have 1 thread, respectively.
💪 R1 runs on NT1 and NT2 (M:N scheduler)

Quoted from RubyKaigi 2022 slides

# Handle managed blocking operations




Start status:
Ruby threads RT1, RT2, RT3 are there
TRQ (Thread Ready Queue) is [RT2, RT2]

❓ RT1 can be scheduled early

# Evaluation
# Ring example

| | Time (sec) |
|---|---|
| Threads (master) | 969.55 |
| Threads (MaNy) | 9.20 |
| Ractors (master) | 166.52 |
| Ractors (MaNy) | 14.22 |
| Ractors (MaNy, MAX_PROC=1) | 7.38 |

x 105.4

x 11.7

x 22.6

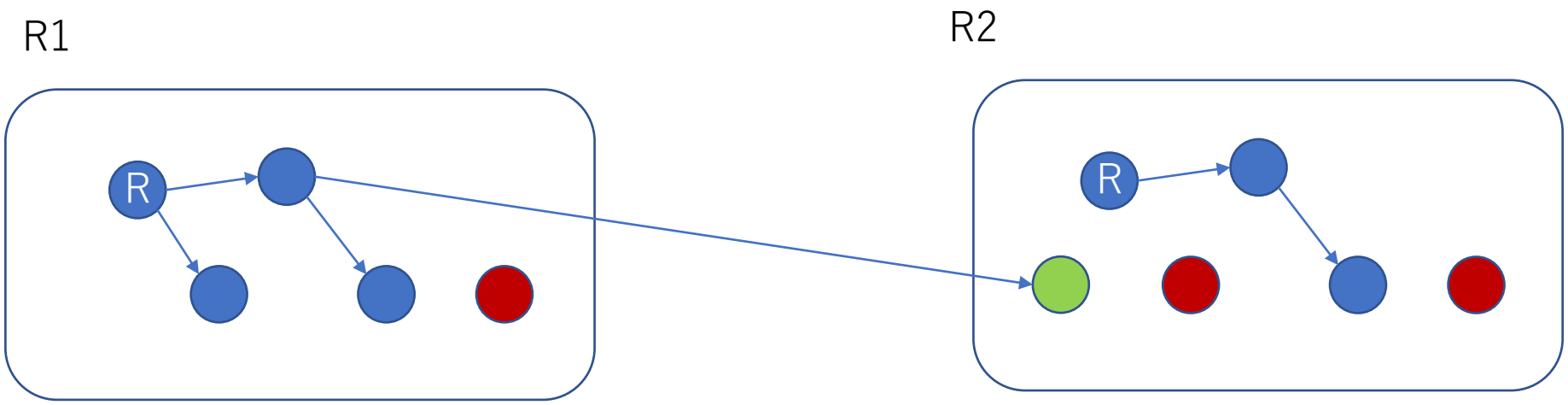
📌Making 1 ring by **10,000** threads/ractors and **1,000** times message passings = **10M** passings
📌Time of making threads/ractors is excluded.
📌Benchmark code: https://gist.github.com/ko1/ac325a785ae292540bd99f141ad55383

# Further performance improvement
# Ractor local GC

- Problem is “There are several shared shareable objects” between ractors

→ Distributed GC (with a few whole GC)

Ractor local GC is ongoing project with GSoC 2022 contributor Rohit Menon



分散GCはやってません！
（鋭意開発中）

# 進捗

- とにかく取り込めるように
  - メイン Ractor では、デフォルトで無効に
  - Linux (epoll使える環境) 以外でも使えるように
- なるべくコアの切り替えをしなくても済むように変更
  - M:N threads では同じ Ractor でもRubyスレッド切り替えでコアの遷移が発生 ⇒ これを起こさないようにスケジューラを書き換え

# TLS 互換性問題

- TLS (Thread local storage)
  - ネイティブスレッドごとに固有の値を設定可能
- TLSを利用しているライブラリを利用している拡張ライブラリで問題がでる可能性
  - NT1 で実行している RT1 が NT2 で実行再開（普通にある）
    ⇒ NT1で設定したTLSが、NT2では入っていない（当然）、困る！
  - Go では runtime.LockOSThread() でネイティブスレッドを固定
  - Ruby にも要ると思う
    - Thread#lock_native_thread() かなぁ（Thread#lock_native_thread? いる？）
    - Ractor#lock_native_thread() もいるかなぁ
- 実際問題になるのか…？
  - errno (TLS) はちょっと問題 ⇒ Ruby では無理やりの解決策
  - Fiber scheduler で問題になるはずだけど、ならなかったのでそんな問題起こらないんでは…？

# 制御用環境変数

- RUBY_NM_THREADS=1
  - メインRactorではデフォルト無効、つまり1:1 スレッド（変わらず）
  - この環境変数で、メインRactorでもMNスレッドに
    - 1 Ractor 利用は 1:Nスレッド（Ruby 1.8 時代と似ている）
    - メインThreadだけは特別にネイティブスレッド占有
  - メイン以外のRactorでは、MNスレッド
- RUBY_MAX_PROC=n
  - M:Nスレッドで利用するネイティブスレッドの数
  - デフォルトは決め打ちで 8 個（CPU数とかにしておきたい）
  - ちなみに、n 個以上利用します
    - メインThread、Timer ネイティブスレッドで +2
    - NTをロックしているRubyスレッド（ブロッキング処理中）

# Future work

- PR 書く
- バグをつぶす（GitHub Actions 上でしか再現しないバグ）
- MN を「利用しない」方法の整理
- kqueue 対応（macOS）、Windows 対応（… はしないかな）
- 今は完全なラウンドロビンスケジューラ→もっと賢く
- Ractor用分散GC

# ご清聴ありがとうございました

クックパッドで6年半やってきた仕事のまとめになります。
2018年にRubyConf (Los angels) のまつもとさんの部屋で
長いこと議論したことがやっと実現できました。