WORKING with RUBY THREADS

Jesse Storimer

Copyright (C) 2013 Jesse Storimer.

This book is dedicated to Sara, Inara, and Ora, who make it all worthwhile.

Contents

Introduction 10

My story 10

Why care? 11

The promise of multi-threading 12

What to expect 13

Which version of Ruby is used? 14

You're Always in a Thread 15

Threads of Execution 17

Shared address space 17

Native threads 21

Non-deterministic context switching 22

Why is this so hard? 27

Lifecycle of a Thread 29

require 'thread' 29

- Thread.new 29
- Thread#join 30
- Thread#value 32
- Thread#status 33
- Thread.stop 34
- Thread.pass 35
- Avoid Thread#raise 35
- Avoid Thread#kill 37
- Supported across implementations 37

Concurrent != Parallel 38

- An illustrative example 39
- You can't guarantee anything will be parallel 41
- The relevance 42

The GIL and MRI 43

- The global lock 43
- An inside look at MRI 44
- The special case: blocking IO 46

- Why? 48
- Misconceptions 50

Real Parallel Threading with JRuby and Rubinius 54

- Proof 54
- But...don't they need a GIL? 57

How Many Threads Are Too Many? 59

- ALL the threads 59
- IO-bound 61
- CPU-bound 66
- So... how many should you use? 71

Thread safety 72

- What's really at stake? 72
- The computer is oblivious 77
- Is anything thread-safe by default? 77

Protecting Data with Mutexes 81

- Mutual exclusion 81
- The contract 84

Making key operations atomic 84
Mutexes and memory visibility 89
Mutex performance 91
The dreaded deadlock 94
Signaling Threads with Condition Variables 100
The API by example 100
Broadcast 105
Thread-safe Data Structures 106
Implementing a thread-safe, blocking queue 106
Queue, from the standard lib 110
Array and Hash 112
Immutable data structures 113
Writing Thread-safe Code 114
Avoid mutating globals 114
Create more objects, rather than sharing one 119
Thread-locals 120
Resource pools 121

Avoid lazy loading 121
Prefer data structures over mutexes 122
Finding bugs 123
Thread-safety on Rails 125
Gem dependencies 125
The request is the boundary 126
Wrap Your Threads in an Abstraction 128
Single level of abstraction 128
Actor model 133
How Sidekiq Uses Celluloid 138
Into the source 140
fetch 140
assign 142
Wrap-up 144
Puma's Thread Pool Implementation 146
A what now? 146
The whole thing 147

In bits 149
Wrap-up 154
Closing 156
Ruby concurrency doesn't suck 157
Appendix: Atomic Compare-and-set Operations 159
Overview 160
Code-driven example 161
Benchmark 166
CAS as a primitive 168
Appendix: Thread-safety and Immutability 170
Immutable Ruby objects 171
Integrating immutability 173
Wrap up 177