Concurrency in Ruby

Las Vegas Ruby Meetup

2012-10-10

David Grayson

Basic Concurrency with a loop

```
task1 = ...
task2 = ...
task3 = ...

while true
  if task1.needs_attention?
    task1.service
  end

  if task2.needs_attention?
    task2.service
  end

  if task3.needs_attention?
    task3.service
  end
end
```

Threads

- Keeps track of a position in the code
- Main thread
- Additional threads

Thread Example

```
require 'net/http'

pages = %w( www.rubycentral.com www.awl.com
  www.pragmaticprogrammer.com )

threads = []

pages.each do |page|
  threads << Thread.new do
    print "Fetching: #{page}\n"
    resp, data = Net::HTTP.get_response(page, '/')
    print "Got #{page}: #{resp.message}\n"
  end
end

threads.each do |thread|
  thread.join
end
```

```
Fetching: www.pragmaticprogrammer.com
Fetching: www.awl.com
Fetching: www.rubycentral.com
Got www.awl.com: Moved Temporarily
Got www.rubycentral.com: OK
Got www.pragmaticprogrammer.com: OK
```

About Threads

- Threads belong to a process
- Share memory with other threads
- MRI: user-space threads
- Jruby and Rubinius: OS threads

Visualizing threads

Basic Thread operations

- Thread.start { … }
- Thread.current
- Kernel#sleep
- stop
- wakeup and run
- kill/terminate/exit
- raise
- join
- value
- status
- Thread-local variables

Implementing Timeout

```ruby
module Timeout
  class Error < RuntimeError
  end

  def timeout(seconds)
    work_thread = Thread.current
    timeout_thread = Thread.start do
      sleep seconds
      work_thread.raise Error
    end
    yield  # perform the work
  ensure
    timeout_thread.kill
    timeout_thread.join # make sure it's dead.
  end

  module_function :timeout
end

Timeout::timeout(20) do
  puts "OK 1"
end
```

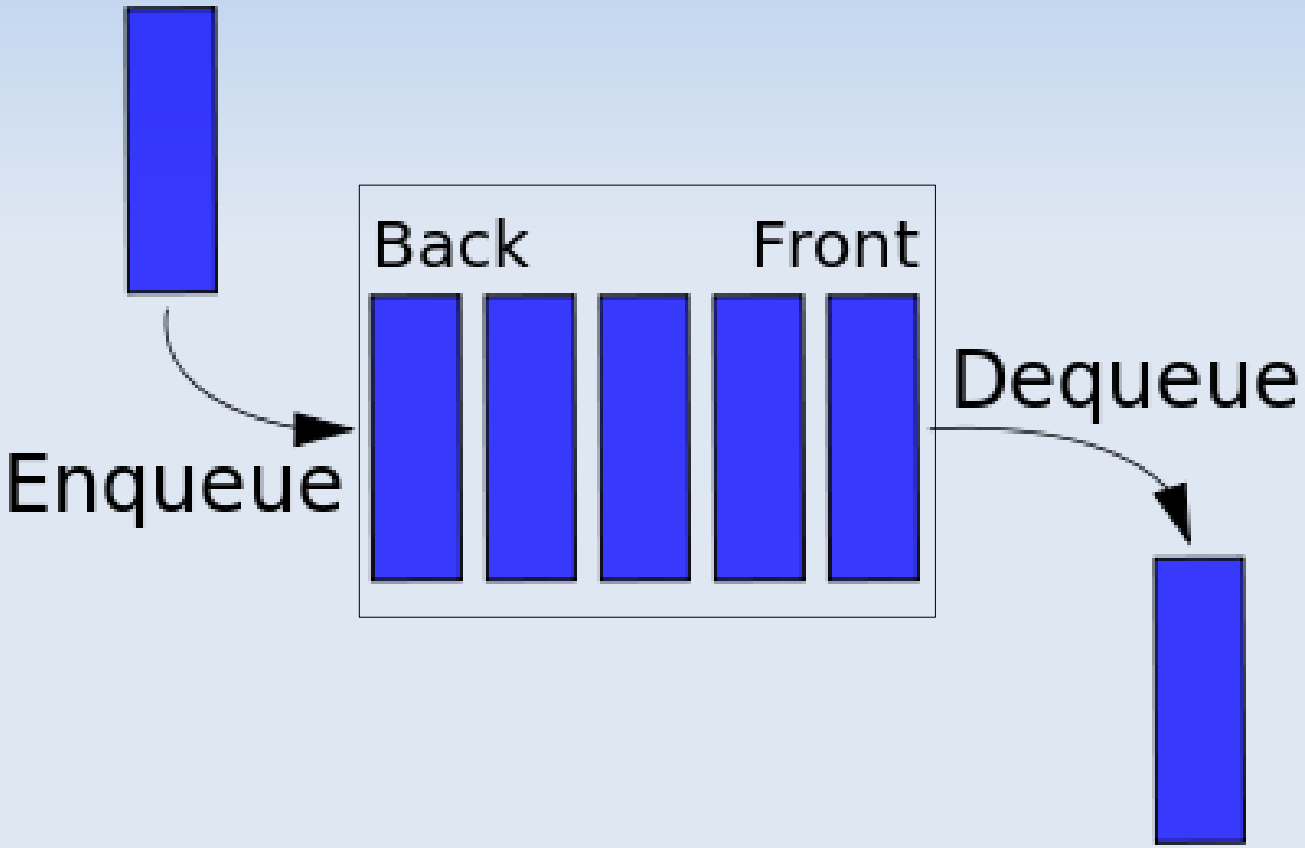
Data Safety

Mutex

Mutual **Ex**clusion

ConditionVariable

Queue



Celluloid / Actors

"I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages"

--Alan Kay, creator of Smalltalk, on the meaning of "object oriented programming"

Drawbacks of Celluloid

- Strange new behavior of ruby objects
- Unclear which objects should be actors
- Hard to step through in a debugger?
- Hard to get a good stack trace?
- Overhead?
- Methods can still access same data concurrently by default!

Conclusion

- **Main tools of concurrency:**
  - **Thread**
  - **Mutex**
- Helper classes written in Ruby:
  - ConditionVariable
  - Queue
- Other concurrency topics:
  - Reactors (eventmachine)
  - Actors (Celluloid)
  - Fibers