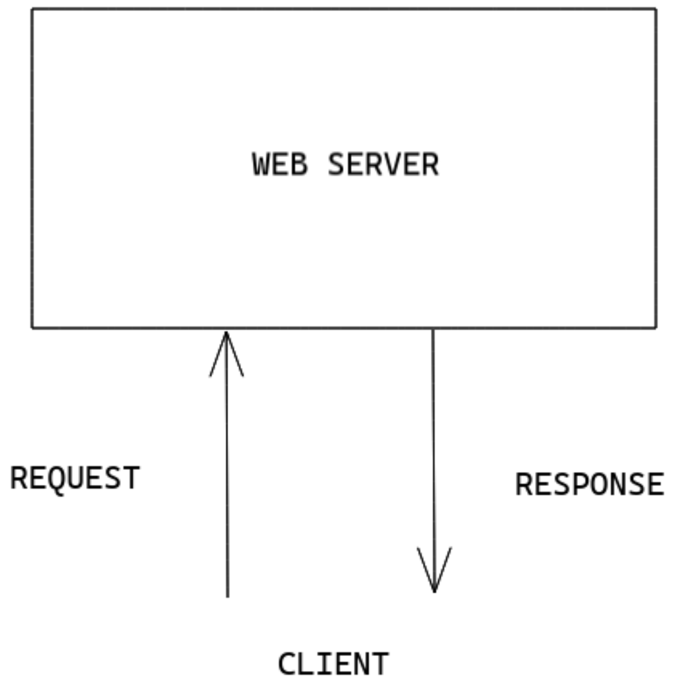
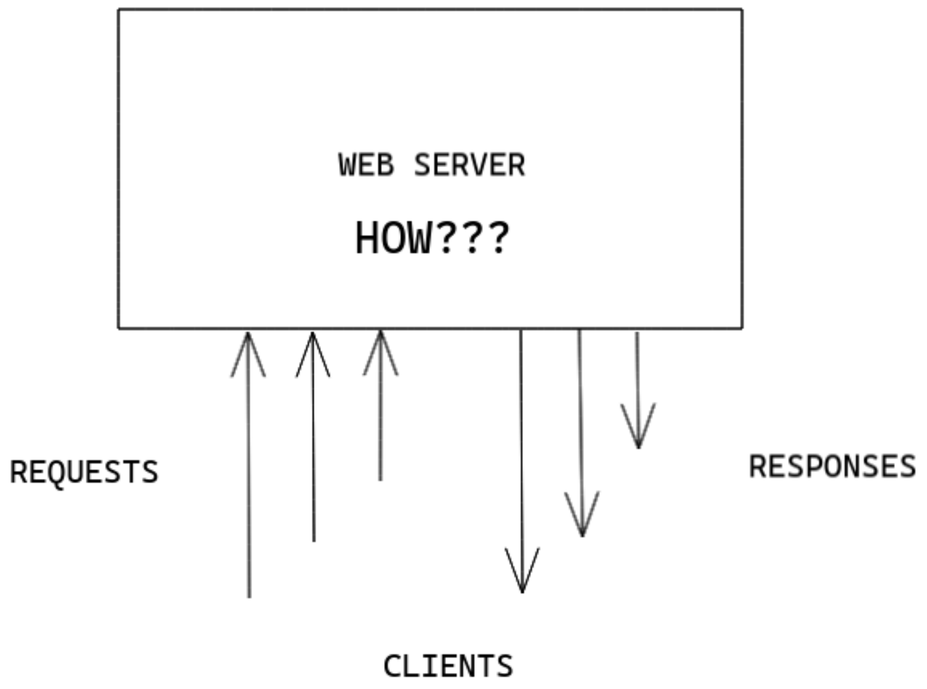
Concurrent programming in Ruby

How a web server is working?



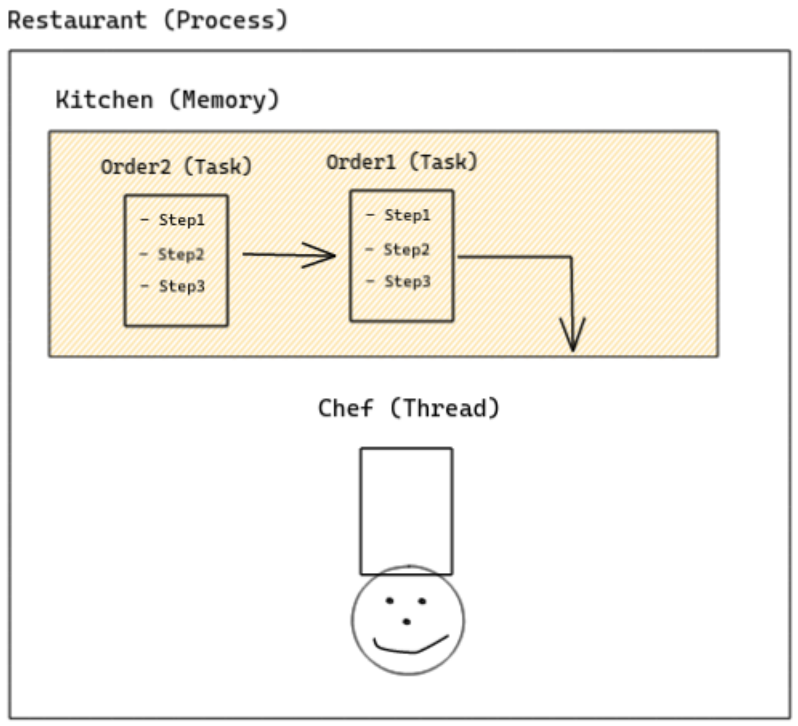
How a web server is REALLY working?



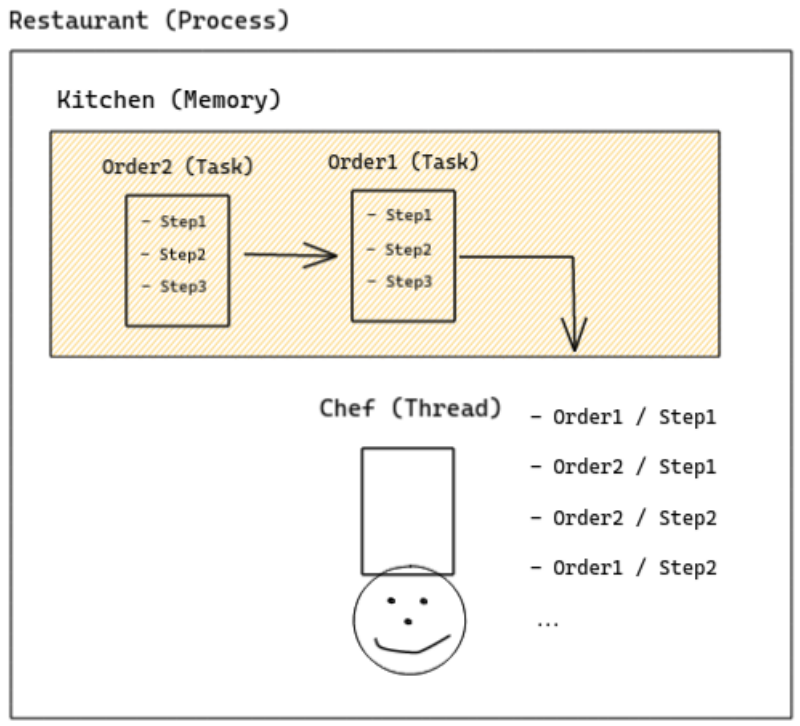
What is concurrency?

Let's take a metaphor...

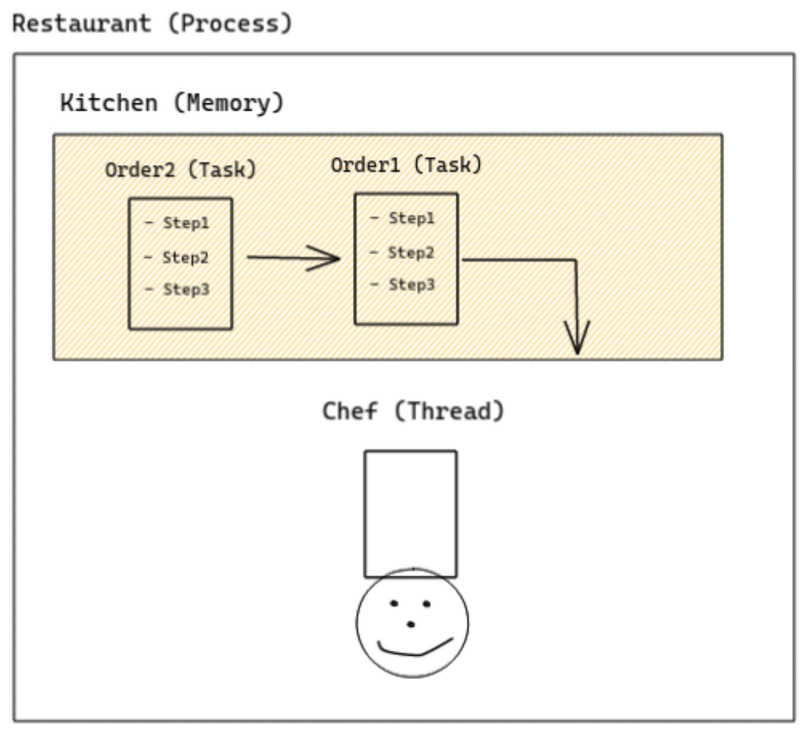
... a Restaurant 👨‍🍳

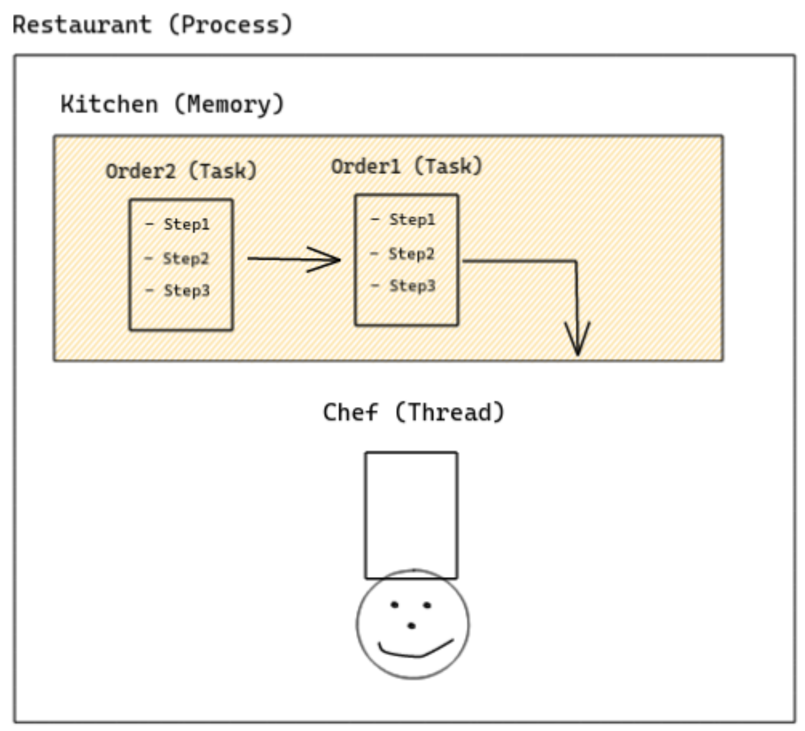


Asynchronicity

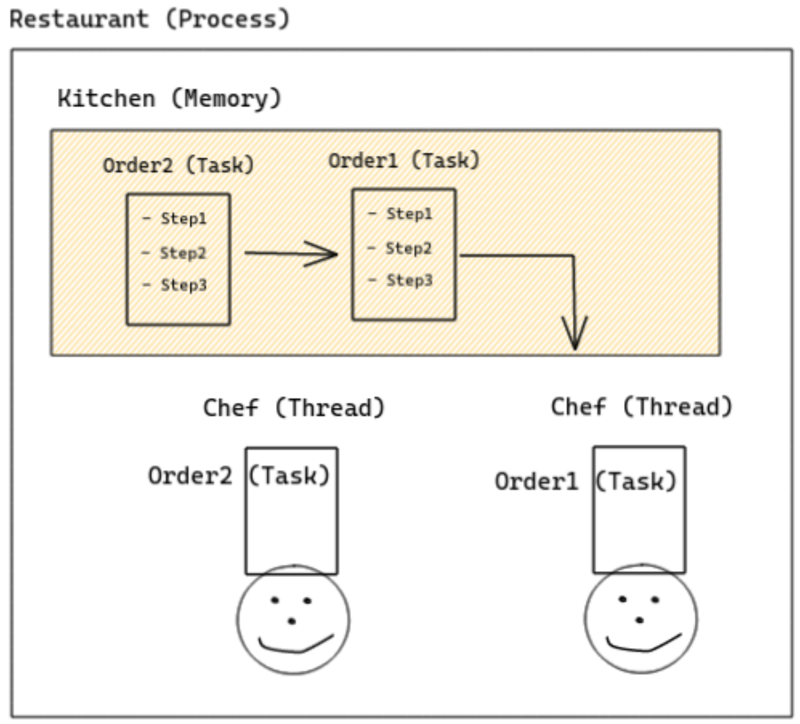


Multi-processing

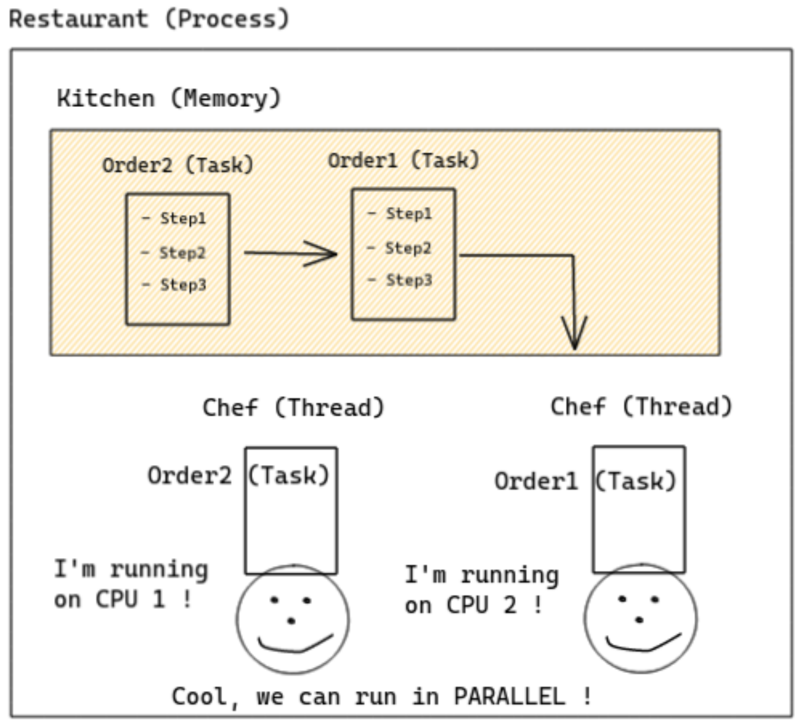




Multi-threading



Parallelism



Takeaways about concurrency

- A program is concurrent when various code sequences run simultaneously
- Concurrency != parallelism
  - A program is parallel when you have multiple CPUs
  - Concurrency is about **dealing with** lots of things at once
  - Parallelism is about **doing** lots of things at once
- There are various implementations of concurrency: multi-threading is one of them

Takeaways about multi-threading

- A process is a running instance of a program
- A thread is a sequence of instructions inside a process
- All threads inside a process share resources such as some memory
- Sharing resources makes it lighter than multi-processing
- But it can leads to thread nightmares 😴😱

Multi-threading in Ruby

Our first multi-threaded program: count from 1 to n

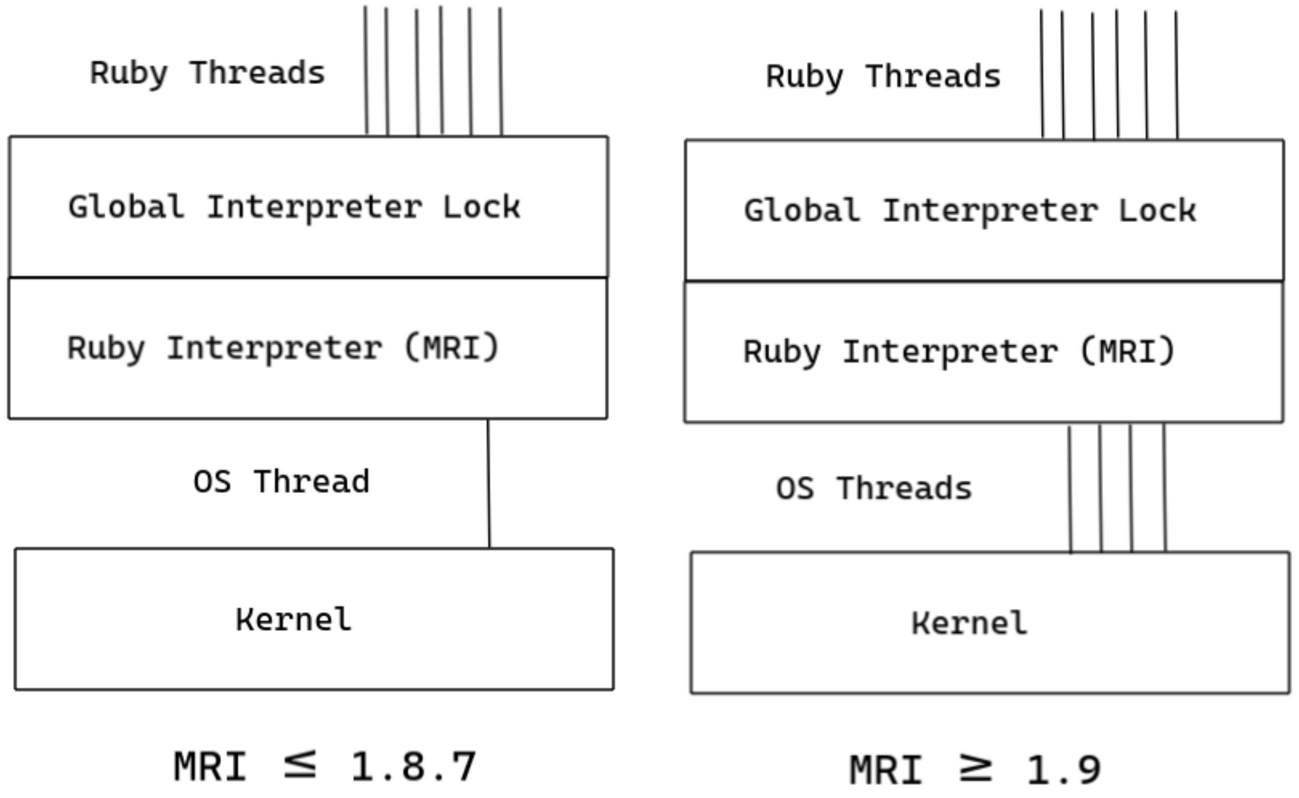
Takeaways

- All threads terminate when the `main` thread terminates
- More threads **doesn't always** lead to more speed
- More threads **can** lead to more speed, especially if they have to wait (I/O)
- When there is I/O you can enter in the rabbit hole of:
    - Race conditions
    - Deadlocks
    - Starvations
    - ...

I have multiple CPUs

Is Ruby concurrent or parallel? 🤔

Ruby Global Interpreter Lock



The GIL is our friend...

- It increases speed of single-threaded programs
- It prevents thread nightmares by executing only one Ruby thread at a time
- It protects Ruby MRI which relies on C libraries which are not always thread-safe

... but recently, it became a bulky friend

- It prevents us to run our programs in parallel and use our full CPU capacity
- It slows down multi-thread programs with few I/O
- It's only present in MRI implementation (cRuby) and not in jRuby for example

Ruby 3: to parallelism and beyond!

Actor design pattern

- An actor is like a thread which doesn't share memory with the other threads
- Actors don't communicate by sharing state
- They share states by communicating to each other

Ractor: Ruby implementation of the Actor design pattern

- Multiple running Ractors in an interpreter process
- Limited object sharing
- Two-types communication between Ractors
- Copy & Move semantics to send message

Our first multi-ractored program: Fibonacci sequence

What is the link with RoR web development?

We already use concurrent programming everyday: `puma`

We already use concurrent programming everyday: `sidekiq`

Conclusion

- Concurrent programming is something we all use, without thinking about it mostly
- Ruby is already tooled for multi-threaded concurrent programming
- But not (yet) for parallel programming
- The implementation of Ractor in Ruby 3 is a hot topic
- Its future use in puma or sidekiq could be a big lever to improve scalability of Ruby on Rails web applications

References

- Multithreading Code - Computerphile
- Restaurant, Kitchen, and Cook Analogy
- Nickel City Ruby 2014- Concurrency for !Dummies (Who Don't Get It (...Yet))
- Ractor report / Koichi Sasada
- Ractor: a proposal for new concurrent abstraction without thread-safety issues

Thank you!