Parallelism in Python

Rounak Vyas

Next Tech Lab
SRM Institute of Science and Technology

About Me

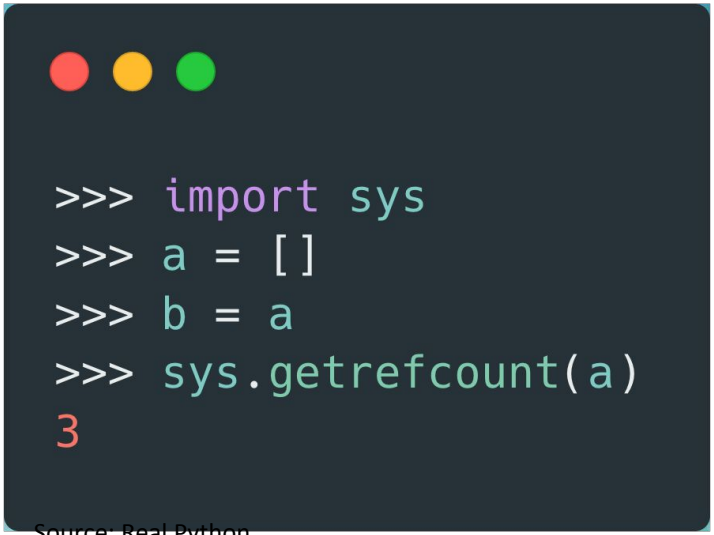
- Final year CS student
- Using python for the last 3 years.
- Student Researcher at Next Tech Lab

Overview

- Python has enjoyed a decade of usage in industry and academia.
- Popular abstractions to scientific computing, AI/ML, etc.
- Yet, has a bad rep for its parallel processing capabilities.

Is multi-threading a scam in Python?

How the interpreter works



Source: Real Python

Python uses reference counting for memory management.

The reference count variable needs protection from race conditions.

How the interpreter works

This count variable can be kept safe by adding locks to all data structures.

But, adding a lock to each object means multiple locks resulting in deadlocks/dec in performance.

Global Interpreter Lock (GIL)

- A mutex (or a lock).
- Allows only one thread to hold the control of the Python interpreter.

Impact of GIL on multi-threaded programs

```
COUNT = 50000000

def countdown(n):
    while n>0:
        n -= 1

countdown(COUNT)
```

Source: Real Python

Single thread ~ 3.60 secs

Impact of GIL on multi-threaded programs

```
COUNT = 50000000

def countdown(n):
    while n>0:
        n -= 1

t1 = Thread(target=countdown, args=(COUNT//2,))
t2 = Thread(target=countdown, args=(COUNT//2,))
```

Source: Real Python

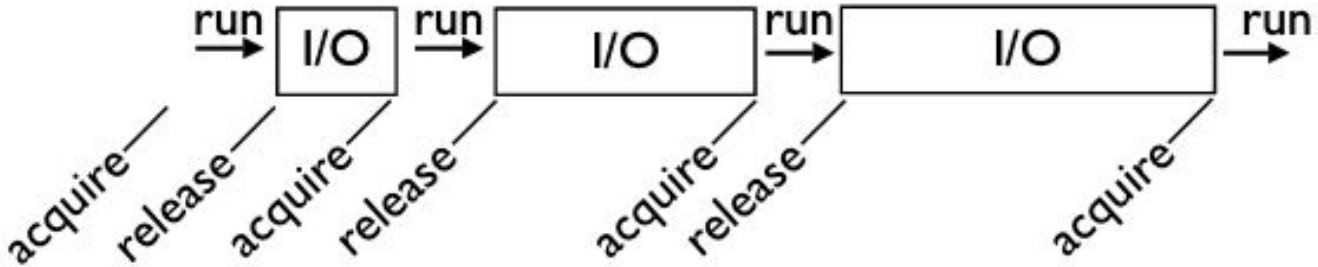
Multiple threads ~ 3.66 secs

(Overhead)

When is GIL not a problem?

I/O Bound Tasks: Everything that blocks the current thread while not consuming much CPU.

I/O Bound Tasks:



Source: David Beazely slides

When it is a problem?

CPU Bound Tasks: Tasks that mostly consume CPU time, like heavy computations or moving lots of data around in-memory (sorting, shuffling)

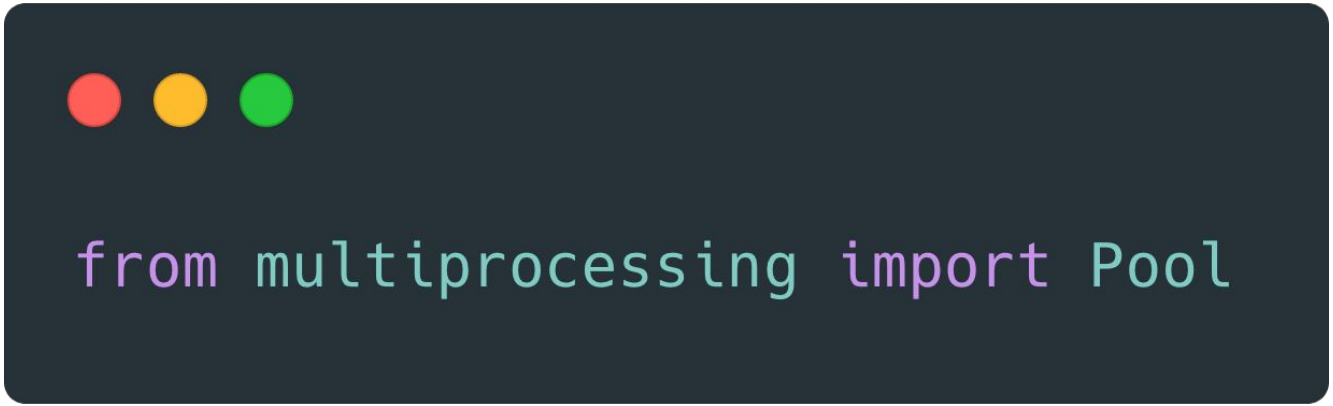
Basically,

So, is it possible to inject parallelism in CPU bound tasks?

Introducing: Multi-Processing

Each Python process gets its own Python interpreter and memory space so the GIL won’t be a problem.

Introducing: Multi-Processing



Real Life Example

Thumbnailing thousands of images.

A common *CPU bound task* for someone working on vision, image processing, etc.

Real Life Example

```
for image in images:
  create_thumbnail(image)
```

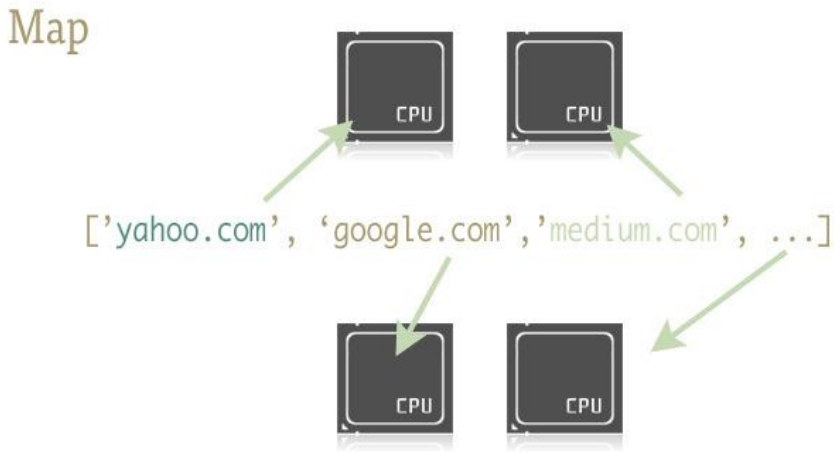
~27.9 seconds to process **6000 Images**

Real Life Example

```
pool = Pool() #default: no. of cores in the system
results = pool.map(create_thumbnail, images)
pool.close()
pool.join()
```

~5.6 seconds!

How Map works?



Bonus.....

Injecting parallelism in IO bound programs

```
from multiprocessing.dummy import Pool as ThreadPool
```

Real Life Example

Retrieving multiple web pages using *urlopen()*.

Real Life Example

```
pool = ThreadPool(4)
results = pool.map(urllib2.urlopen, urls)
pool.close()
pool.join()
```

Real Life Example: 15 URLs

```
Single thread:  14.4 Seconds
       4 Pool:   3.1 Seconds
       8 Pool:   1.4 Seconds
      13 Pool:   1.3 Seconds
```

“There’s more than one way to do it”

Feel free to reach out!

Thank You ！


rounakvyas@outlook.com


itsron717


itsron143

https://rounakvyas.me