【Python教學】淺談 GIL & Thread-safe & Atomic operation

BY MAX ALL POSTS、PYTHON 基礎教學、PYTHON 爬蟲教學 22 6 月, 2023

本篇整理了關於 Python 為什麼會有 GIL 的出現、thread-safe 問題探討、以及 GIL 切換時機、和確保 thread-safe 的原子操作概念 (atomic operation)，此篇未來會持續更新，希望對在了解 GIL 的你有幫助～

Table

一. 為什麼會有 GIL 的出現？

> In CPython, the global interpreter lock, or GIL, is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once. This lock is necessary mainly because CPython's memory management is not thread-safe. (However, since the GIL exists, other features have grown to depend on the guarantees that it enforces.)
>
> WIKI.PYTHON.ORG

避免在在執行 multiple threads 時，CPython memory 會有 thread-safe 的問題，所以在 Python Source Code 直譯成 bytecodes 時增加 GIL (Global Interpreter Lock) 的全域鎖。也就是說 GIL 可以用於確保在 Python 運行時僅運行一個 Thread 來保證 Thread-safe。

參考文章：wiki.python.org

二. 切換 thread 的時機？

當一個執行緒沒有直譯超過 1000 個 bytecode 指令 (Python 2)，或是超過 5 ms (Python 3.2 後) 的時候，執行緒將會釋放 GIL 來讓其他執行緒使用。所以 GIL 保證的 Thread-safe 是 Bytecode 而不是 Python Code。

參考文章：深入 GIL: 如何寫出快速且 thread-safe 的 Python

三. 什麼是 thread-safe？

當多執行緒同時運行，並對同一資源進行讀寫操作的修改時，必須保證其執行緒與執行緒間不會發生衝突，和數據修改不會發生錯誤，稱為 thread-safe。

而了解了 thread 的切換時機和 thread-safe 後，如何避免執行緒執行到一半就被其他執行緒，就要討論原子操作。

四. 什麼是原子操作 (atomic operation)？

同上面所述的 GIL 保證的 Thread-safe 是在 Bytecode 層而不是 Python Code。所以能確保的是每行 Bytecode 都會被運行完成，而多行 Bytecode 時則有被中斷切換執行緒的可能性。

1. 舉個例子：
運行 sort() 函式

```
import dis

this_is_list = []

def list_sort():
    this_is_list.sort()
    return ‘ok’

dis.dis(list_sort)
```

輸出結果：

```
>>> 0 LOAD_GLOBAL      0 (this_is_list)
>>> 2 LOAD_METHOD      1 (sort)
>>> 4 CALL_METHOD      0
>>> 6 POP_TOP
```

所以 sort 本身是單一個 bytecode，這代表在執行完 sort 前是沒有機會被中斷而釋放 GIL 的。而這個概念就叫做原子操作 (atomic operation)，也就是 “原子是最小的、不可分割的最小個體” 的意義。

2. 再舉個例子：
運行 append() 函式

```
import dis

this_is_list = []


def list_append():
    this_is_list.append('text')
    return 'ok'

dis.dis(list_append)
```

輸出結果：

```
0 LOAD_GLOBAL     0 (this_is_list)
2 LOAD_METHOD     1 (append)
4 LOAD_CONST      1 ('text')
```

```
4  6 CALL_METHOD     1
5  8 POP_TOP
```

可以看到 append 函式的輸出中，比剛剛運行 sort 還多了一行 bytecode (4 LOAD_CONST 1 ('text'))，也就是說有可能在運行到一半時，就被中斷換其他執行緒運行，所以 append 函式就不是一個原子操作～

五. 非原子操作，如何避免 Race Condition

如果是非原子操作 (atomic operation) 的情況下，threading 標準庫內有提供一些方式來防止 race condition 的發生。而 Lock 和 RLock 其中兩個基本工具。

- Lock：Lock 是一個像大廳通行證的對象。一次只有一個 thread 可以擁有 Lock。基本功能是 `.acquire()` 和 `.release()` 或使用 `with self._lock`
- RLock：避免 Deadlock 發生，它允許一個 thread 在調用 `.release()` 之前多次 `.acquire()` ，也就是鎖中鎖的概念，可以遞迴的鎖。