حوری دهش 9821413

GIL

GIL یا همون Global Interpreter Lock، یک قفل همگانی در زبان برنامه‌نویسی پایتون است که این قفل به ما این اطمینال رو میده که تنها یک ترد در هر زمان در هر مرحله اجرا، انجام میشود یعنی هنگام اجرای کد پایتون تنها یک ترد می تواند به طور همزمان دستورات CPython (پیاده سازی استاندارد پایتون) را پیاده سازی کند و چند ترد را در یک برنامه نمی توان به طور همزمان اجرا کرد در نهایت برای برنامه هایی که همزمانی بالا یا پردازش‌های موازی نیاز دارند، ممکن است gil محدودیت‌هایی ایجاد کند.

در حالت کلی مزیت هایی که gil برای ما دارد:

- پیاده سازی gil ساده است و به راحتی به پایتون اضافه شده و این افزایش عملکرد را برای برنامه های تک رشته ای فراهم می کند زیرا فقط یک قفل باید مدیریت شود.
- برای برنامه های I/O bound مناسب است چون در این حالت، اگر یک ترد برای انجام عملیات I/O بلاک شود یا به عبارت دیگر منتظر نتیجه I/O باشد، تردهای دیگر همچنان می‌توانند اجرا شوند.

و معایبی که gil دارد:

- باعث ایجاد محدودیت در همزمانی می شود.
- برای برنامه های cpu bound به خوبی کار نمیکند چون در این حالت، تردی که به عملیات محاسباتی مشغول شده است تا زمان اتمام این محاسبات، gil را در اختیار می‌گیرد و سایر تردها متوقف می‌شوند که این ممکن است باعث افت کارایی شود.

نکته ای که وجود دارد این است که، Scheduling مسئول ترتیب اجرای تردها است و در هر لحظه مشخص می‌کند که کدام ترد در حال حاضر باید اجرا شود

در برنامه‌های پایتون با وجود gil --> Concurrency و Scheduling می‌تواند با از روش‌های مختلفی انجام شود.

همان طور که گفته شد gil باعث محدودیت در همزمانی اجراها در سطح کد Python می‌شود اما به این معنا نیست که امکان همزمانی در سطح سیستم وجود ندارد. دو روش در زیر گفته میشود که امکان همزمانی در پایتون را نشان میدهد:

روش اول --> استفاده از تردها:

در این روش از ماژول threading برای ایجاد تردها استفاده میشود و از انجایی که gil باعث محدودیت در همزمانی در سطح کد پایتون میشود اما تردها می توانند در برنامه های I/O bound مفید باشند به این صورت که در زمان انتظار برای I/O تردهای دیگر میتوانند اجرا شوند.

روش دوم --> استفاده از asyncio:

در این روش برای برنامه های I/O bound می توان از ماژول asyncio استفاده کرد و در زمان انتظار برای I/O به تعداد دیگری از توابع فرصت داده میشود تا اجرا شوند.

مثالی از این روش:

```python
import asyncio

async def test1():
    print("Start test1")
    await asyncio.sleep(1)
    print("End test1")

async def test2():
    print("Start test2")
    await asyncio.sleep(2)
    print("End test2")

asyncio.run(asyncio.gather(test1(), test2()))
```

توضیح کد بالا:

- توابع test2 , test1 به صورت همزمان اجرا میشوند و از async/await برای مدیریت عملیات های I/O استفاده می کنند. همینطور تابع sleep در کد بالا برای نمایش عملیات I/O استفاده شده است.
- تابع asyncio.gather به عنوان یکی از روش های همزمانی در asyncio استفاده شده است این تابع اجازه میدهد تا توابع test2 , test1 همزمان اجرا شوند و انتظار اتمام هر کدام نباشند.
- تابع asyncio.run برای اجرای یک تابع async به عنوان تابع اصلی برنامه استفاده میشود. در اینجا تابع main اجرا می شود که با استفاده از asyncio.gather توابع test2 , test1 را همزمان اجرا می‌کند.

همچنین برای بلاک کردن یک ترد در پایتون از متدهای مرتبط با lock و condition در ماژول threading استفاده می شود.

مقایسه تردینگ در پایتون و زبان C:

1. gil در پایتون باعث محدودیت در همزمانی اجراها، در سطح کد Python می شود که این برای برنامه هایی که cpu bound هستند می تواند یک محدودیت باشد ولی در زبان C ما gil مرتبط با تردها نداریم پس تردها می توانند به طور همزمان اجرا شوند و در نهایت این امکان را به برنامه نویس می‌دهد که بهترین استفاده از منابع سیستم را که قابل ارائه توسط CPU هستند، داشته باشند. در آخر برای برنامه ای که cpu bound است، استفاده از زبان C بدون gil ممکن است یک گزینه بهتر باشد. اما در برنامه های I/O bound، پایتون با استفاده از ابزار هایی مانند asyncio می‌تواند به صورت همزمان و بهینه اجرا شود.
2. در زبان C مدیریت حافظه تردها به عهده برنامه نویس است و او باید مسئولیت ایجاد و از بین بردن تردها را به عهده داشته باشد ولی در پایتون مدیریت حافظه تردها به صورت خودکار انجام می‌شود. Python با استفاده از garbage collector از تردها و منابع مربوط به ان ها به طور خودکار مراقبت می‌کند.
3. در زبان C استفاده از سیستم‌های متفاوت مانند pthreads برای Unix یا Windows API برای ویندوز برای ایجاد و مدیریت تردها رایج است. این کتابخانه ها امکانات مستقلی برای مدیریت تردها ارائه میدهند ولی در پایتون از ماژول threading برای تردینگ استفاده می شود که این ماژول ارائه های بالاتری نسبت به توابع پایه ای ترد در زبان C دارد اما به دلیل وجود gil، تردها در برنامه‌های cpu bound کمک زیادی نمی کنند.