Goodbye GIL: Exploring the Free-threaded mode in Python 3.13

Pycascades 2025

About Me

Adarsh Divakaran

Co-founder - LinkHQ by Digievo Labs

Outline

GIL

- Performance impact
- PEP 703

Free Threaded Python

- Setup & Run

Performance Comparison

- Single threaded vs Multithreaded
- CPU & I/O bound

GIL

- Mutex in CPython preventing multiple native threads from executing Python bytecode simultaneously.
- Ensures only one native thread executes Python bytecode at a time within a single process.
- Simplified CPython's memory management and C extension API.

GIL

The Problem: Impact on Multithreading

- Limits true parallelism for CPU-bound tasks.
- I/O-bound tasks less affected as GIL is released during I/O operations.
- Historically, multiprocessing was the primary way to achieve parallelism in Python for CPU-bound tasks, but with inter-process communication overhead.

PEP 703

PEP 703 - Making the Global Interpreter Lock Optional

- PEP 703 proposes making the Global Interpreter Lock optional in CPython.
- Introduces "free-threaded build" of Python where the GIL is disabled.
- Aims to unlock true parallel execution for Python threads, especially for CPU-bound workloads.

PEP 703

- Experimental in Python 3.13, eventually evolving into an optional (or possibly default-disabled) feature in future releases.
- **Backward Compatibility**: Many C extensions are written assuming the GIL will exist.
- **Performance Overhead**: Free-threaded builds may run single-threaded code slower.

Setting Up Free-Threaded Mode

- Installation Guide: https://py-free-threading.github.io/installing_cpython
- At Build time: Building from source with --disable-gil
- Windows/Macos: Use official installers from Python.org

Setting Up Free-Threaded Mode

- Ubuntu:

sudo add-apt-repository ppa:deadsnakes

sudo apt-get update

sudo apt-get install python3.13-nogil

Creating Virtual Environment

Use the free-threaded executable:

```
python3.13t -m venv .venv
```

Setting Up Free-Threaded Mode

Verifying the set-up

- sys._is_gil_enabled()

- sysconfig. get_config_var("Py_GIL_DISABLED")

Benchmarks

The Changes

- “ The reference implementation changes approximately 15,000 lines of code in CPython and includes mimalloc, which is also approximately 15,000 lines of code. “ - PEP 703

We will compare the execution time of free-threaded interpreter vs GIL enabled default interpreter of Python 3.13.

Single Threaded Program

DEMO - Single Threaded Program

- We will time the execution of a simple math operation.
- Compare default vs free-threaded interpreter performance.

Single Threaded Program

Conclusion:

- GIL Limits true parallelism for multithreaded code

For single threaded programs, free-threaded mode has a negative impact on performance

CPU Bounded Program

DEMO - Gil enabled vs disabled

- Perform computationally intensive operation - Single threaded vs multithreaded
- Benchmark 3.13 vs 3.13t performance

IO Bounded Program

DEMO - Gil enabled vs disabled

- Perform I/O operation (writing to a file / network calles, etc) - Single threaded vs multithreaded
- Benchmark 3.13 vs 3.13t performance

WSGI Example - Flask App

DEMO - Gil enabled vs disabled

- 2 API Endpoints - 1 running I/O bound & other CPU bound
- Benchmark 3.13 vs 3.13t performance

File	GIL Enabled	Free Threaded
1_single_threaded.py	~2.8s	~3.5s
2_io_bound_single_thread	~3.4	~3.4
2_io_bound_multithread	~3.4	~3.6
3_cpu_bound_single_thread	~3.2	~3.2
3_cpu_bound_multithread	~3.05	~1.12
flask_app_cpu_bound	55 req/30s	141 reqs/30s
flask_app_io_bound	128 req/30s	109 req/30s

Migration Checklist

- Great for CPU bound Tasks + Threading
- Great for scenarios where parallelisation is needed - but multiprocessing is too complex / not suitable
- Need to check third party library compatibility with free-threaded mode

The Future - PEP 703

1. In 2024, CPython 3.13 is released with support for a `--disable-gil` build time flag. There are two ABIs for CPython, one with the GIL and one without. Extension authors target both ABIs.
2. After 2–3 releases, (i.e., in 2026–2027), CPython is released with the GIL controlled by a runtime environment variable or flag. The GIL is enabled by default. There is only a single ABI.
3. After another 2–3 release (i.e., 2028–2030), CPython switches to the GIL being disabled by default. The GIL can still be enabled at runtime via an environment variable or command line flag.

The Future - PEP 703

The changes proposed in the PEP will increase execution overhead for `--disable-gil` builds compared to Python builds with the GIL. In other words, it will have slower single-threaded performance. There are some possible optimizations to reduce execution overhead, especially for `--disable-gil` builds that only use a single thread. These may be worthwhile if a longer term goal is to have a single build mode, but the choice of optimizations and their trade-offs remain an open issue.

In future, performance penalties will be reduced for free threaded interpreter.

Conclusion

- Great improvement seen in multithreaded CPU-bound programs. A new concurrency mode is unlocked for CPU bound programs without overhead of inter process communication.

- For single-threaded programs - Go with GIL-Disabled default interpreter
- For programs using multiprocessing - Go with GIL-Disabled default interpreter

Thank You

linkhq.co/adarsh