Философия полезности asyncio

Для асинхронного программирования в Python старше версии 3.5 используется основная библиотека Asyncio - на сегодня это лучший способ освоить асинхронность. Удобно то, что она полностью встроена в ядро.

Асинхронный режим в сравнении с многопоточностью имеет ряд преимуществ:

- Функции асинхронности намного легче потоков. Это значит, что сотни и тысячи асинхронных операций, выполняемых одновременно, будут расходовать гораздо меньше ресурсов, чем сотни и тысячи потоков.
- По сравнению с потоками асинхронными операциями намного легче управлять. Например, их можно отменять, задействовав объект **Task - asyncio.create_task()**.
- В цикле асинхронных событий задачи выполняются в одном потоке, их проще понимать, контролировать и отслеживать.

Фактически асинхронность идет рука об руку с многопроцессорностью в Python. Есть возможность использовать **asyncio.run_in_executor()** для задач, которые интенсивно используют центральный процессор. **Также асинхронность решает проблемы потоков:**

1. Библиотека Asyncio осуществляет переключение контекста на уровне программы, а не процессора. Используется цикл событий.

2. Отсутствие состояние гонки (Race Condition) . Поскольку с Asyncio переключение контекста осуществляется в заранее созданных точках, код не испытывает проблему «гонки». Запускается одна сопрограмма, которая переключается только в заданных вами точках.

3. Гораздо меньшее ресурсное голодание. Конечно, в Asyncio имеется пул потоков, который может влиять на расходование ресурсов (запуск большого количества процессов). Но тем не менее запуск сопрограмм в одном потоке задействует гораздо меньше памяти на выполнение процессов.

4 Взаимная блокировка. Отсутствие гонки потоков практически сводит к нулю различного рода блокировки.

Есть и небольшие недостатки использования библиотеки Asyncio. Во избежание блокировки цикла событий и временных потерь на выполнении асинхронных функций весь код также должен быть *асинхронным*.

Тем не менее мы можем наблюдать динамику увеличения количества асинхронных библиотек и специального программного обеспечения, предоставляющего асинхронные неблокирующие версии баз данных, сетевых протоколов. Например, репозиторий aio-libs — набор библиотек, созданный на основе Asyncio, в котором представлена асинхронная библиотека aiohttp для веб-доступа. Или же в каталоге пакетов Python также много библиотек с async.
Даже такие монстры как Facebook, Twitter, фреймворк React Native и база данных RocksDB используют асинхронность.

Смог бы, например, Twitter быстро обрабатывать миллиарды сеансов в день без асинхронного программирования?

Лучший способ научиться асинхронному программированию — посмотреть, как применяют его на практике другие программисты.