Urban University

ДИПЛОМНАЯ РАБОТА

**Сравнение различных подходов к реализации асинхронного программирования: asyncio, threading и multiprocessing**

**Четвериков Артем**

**г. Уфа, 2024**

**Параллелизм** заключается в выполнении нескольких операций одновременно.

**Многопроцессорность** — это способ реализации параллелизма, который предполагает распределение задач между центральными процессорами (ЦП) или ядрами компьютера. Многопроцессорность хорошо подходит для задач, связанных с ЦП: к этой категории обычно относятся тесно связанные for циклы и математические вычисления.

**Параллельная обработка**— это более широкое понятие, чем параллелизм. Оно предполагает, что несколько задач могут выполняться одновременно. (Есть поговорка, что параллельная обработка не подразумевает параллелизм.)

**Многопоточность** — это модель параллельного выполнения, при которой несколько потоков по очереди выполняют задачи. Один процесс может содержать несколько потоков. Что важно знать о многопоточности, так это то, что она лучше подходит для задач, связанных с вводом-выводом. В то время как задача, связанная с процессором, характеризуется тем, что ядра компьютера постоянно работают от начала до конца, в задаче, связанной с вводом-выводом, большую часть времени занимает ожидание завершения ввода/вывода.

Подводя итог вышесказанному, можно сказать, что параллелизм включает в себя как многопроцессорность (идеальную для задач, связанных с процессором), так и многопоточность (подходящую для задач, связанных с вводом-выводом). Многопроцессорность — это форма параллелизма, а параллелизм — это конкретный тип (подмножество) параллельной обработки. Стандартная библиотека Python поддерживает оба этих варианта с помощью своих multiprocessing, threading, и concurrent.futures пакетов.

Теперь пришло время добавить в эту смесь нового участника. За последние несколько лет в CPython был более тщательно встроен отдельный дизайн: асинхронный ввод-вывод, доступный через пакет стандартной библиотеки asyncio и новые ключевые слова async и await.

В документации Python этот asyncio пакет позиционируется как библиотека для написания параллельного кода. Однако асинхронный ввод-вывод не является ни многопоточностью, ни мультипроцессингом. Он не основан ни на том, ни на другом.

На самом деле асинхронный ввод-вывод — это однопоточная, однопроцессная конструкция: она использует кооперативную многозадачность, термин, который вы расшифруете к концу этого руководства. Другими словами, асинхронный ввод-вывод создаёт ощущение параллелизма, несмотря на использование одного потока в одном процессе. Корутины (основная функция асинхронного ввода-вывода) могут планироваться одновременно, но они не являются по своей сути параллельными.

Повторюсь, асинхронный ввод-вывод — это стиль параллельного программирования, но не параллелизм. Он больше похож на многопоточность, чем на многопроцессорность, но сильно отличается от них обоих и является самостоятельным элементом в наборе инструментов для параллельного программирования.

Асинхронные процедуры могут «приостанавливаться» в ожидании конечного результата и тем временем запускать другие процедуры.

Асинхронный код с помощью описанного выше механизма обеспечивает параллельное выполнение. Другими словами, асинхронный код имитирует параллельное выполнение.

Асинхронный ввод-вывод позволяет сократить время ожидания, в течение которого функции в противном случае блокировались бы, и даёт возможность другим функциям выполняться во время простоя.

Структура программы высокого уровня будет выглядеть следующим образом:

- Считывает последовательность URL-адресов из локального файла, urls.txt.

- Отправьте GET-запросы по URL-адресам и декодируйте полученный контент. Если это не сработает, остановитесь на этом URL-адресе.

- Ищите URL-адреса в href тегах в HTML-формате ответов.

- Запишите результаты в foundurls.txt.

Выполняйте все вышеперечисленное как можно более асинхронно и одновременно. (Используйте aiohttp для запросов и aiofiles для добавления файлов. Это два основных примера ввода-вывода, которые хорошо подходят для модели асинхронного ввода-вывода.)

Вот содержимое urls.txt. Оно не очень большое и содержит в основном сайты с высокой посещаемостью:

Многопоточность также масштабируется менее эффективно, чем асинхронный ввод-вывод, потому что потоки — это системный ресурс с ограниченной доступностью. Создание тысяч потоков приведёт к сбою на многих компьютерах, и я не рекомендую пытаться это сделать. Создание тысяч задач асинхронного ввода-вывода вполне возможно.

Асинхронный ввод-вывод полезен, когда у вас есть несколько задач, связанных с вводом-выводом, в которых в противном случае преобладал бы блокирующий ввод-вывод, например:

-Сетевой ввод-вывод, независимо от того, является ли ваша программа серверной или клиентской

-Бессерверные системы, такие как одноранговая многопользовательская сеть, например групповой чат

-Операции чтения/записи, при которых вы хотите имитировать стиль «запустил и забыл», но при этом не беспокоиться о блокировке того, что вы читаете и записываете