Urban University

ДИПЛОМНАЯ РАБОТА

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

**Автор: Четвериков Артем Васильевич**

**г. Уфа, 2024**

Оглавление

[Цель работы 3](#__RefHeading___Toc1055_4131462224)

[Введение 4](#__RefHeading___Toc845_4131462224)

[Обоснование выбора темы 4](#__RefHeading___Toc1059_4131462224)

[Определение цели и задач исследования 4](#__RefHeading___Toc1063_4131462224)

[Основные понятия и определения 5](#__RefHeading___Toc1065_4131462224)

[Методы и подходы к разработке 7](#__RefHeading___Toc1067_4131462224)

[Обзор популярных библиотек для параллельного программирования на Python 8](#__RefHeading___Toc1069_4131462224)

[Библиотека asyncio 8](#__RefHeading___Toc1071_4131462224)

[Библиотека threading 8](#__RefHeading___Toc1073_4131462224)

[Библиотека multiprocessing 8](#__RefHeading___Toc1075_4131462224)

[Проектирование приложения 9](#__RefHeading___Toc1077_4131462224)

[Анализ и интерпретация результатов 10](#__RefHeading___Toc1079_4131462224)

[Заключение 11](#__RefHeading___Toc1081_4131462224)

# Цель работы

Реализовать асинхронные задачи с использованием asyncio, threading и multiprocessing, сравнить их производительность и уместность для различных типов задач.

# ****Введение****

## Обоснование выбора темы

## Определение цели и задач исследования

# Основные понятия и определения

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. В Python встроен отдельный дизайн: асинхронный ввод-вывод, доступный через пакет стандартной библиотеки asyncio и 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. 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.

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.

Многопоточность 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 это не рекомендуется делать. 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 программа серверной или клиентской

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

- операции чтения/записи, при которых нужно имитировать стиль «запустил и забыл», но при этом не беспокоиться о блокировке того, что читается и записывается

# Методы и подходы к разработке

# Обзор популярных библиотек для параллельного программирования на Python

## Библиотека asyncio

## Библиотека threading

## Библиотека multiprocessing

# Проектирование приложения

# Анализ и интерпретация результатов

# Заключение