**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Домашнее задание

«Параллельное и асинхронное программирования в python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф.ИУ5 |
| Борисова К.А. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г.

**Введение**

В современном мире информационные технологии играют ключевую роль в развитии общества. Они применяются во всех сферах человеческой деятельности, от науки и образования до промышленности и бизнеса. Одной из наиболее популярных и востребованных технологий является программирование. Оно позволяет создавать сложные системы и приложения, которые облегчают нашу жизнь и делают её более удобной и эффективной.

Параллельное и асинхронное программирование — это два подхода к написанию кода, которые позволяют ускорить выполнение задач и повысить производительность программ. В последние годы эти подходы становятся всё более популярными среди разработчиков, так как они позволяют эффективно использовать ресурсы современных многоядерных процессоров и сетевых устройств.

Целью данного реферата является изучение параллельного и асинхронного программирования на Python, их основных принципов и инструментов, а также сравнение этих подходов и определение их преимуществ и недостатков. Для достижения этой цели будут рассмотрены следующие задачи:

* Определение параллельного программирования и его основных инструментов в Python;
* Изучение асинхронного подхода и его отличий от параллельного;
* Сравнение параллельного и асинхронного подходов с точки зрения производительности и эффективности;
* Выявление сценариев, в которых предпочтительно использовать тот или иной подход.

**Параллельное программирование в Python**

Параллельное программирование — это подход, при котором несколько задач выполняются одновременно. Это позволяет ускорить выполнение программы и повысить её производительность. В Python параллельное программирование реализуется с помощью многопоточности и многозадачности.

Многопоточность — это возможность выполнять несколько потоков (подзадач) внутри одной программы одновременно. Каждый поток имеет свой собственный стек вызовов и может выполняться независимо от других потоков. Многопоточность в Python реализуется с помощью модуля threading.

import threading

def my\_function():

print("Hello from thread!")

thread = threading.Thread(target=my\_function)

thread.start()

В этом примере мы создаём новый поток с помощью класса Thread и запускаем его с помощью метода start(). Когда поток будет запущен, он выполнит функцию my\_function(), которая выведет на экран строку «Hello from thread!».

Многозадачность — это способность операционной системы выполнять несколько программ одновременно. В отличие от многопоточности, где все потоки выполняются внутри одной программы, многозадачность позволяет запускать несколько независимых программ одновременно. Многозадачность в Python реализуется с помощью модулей multiprocessing и concurrent.futures.

from multiprocessing import Pool

def square(x):

return x \* x

if \_\_name\_\_ == '\_\_main\_\_':

with Pool(processes=4) as pool:

result = pool.map(square, [1, 2, 3, 4])

print(result)

Этот пример использует модуль multiprocessing для создания пула процессов. Затем мы используем метод map() для вычисления квадратов чисел от 1 до 4. Метод map() принимает два аргумента: функцию, которую нужно применить к каждому элементу списка, и сам список. В данном случае мы передаём функцию square() и список [1, 2, 3, 4]. Метод map() возвращает список результатов, который мы выводим на экран.

**Асинхронное программирование в Python**

Асинхронное программирование — это подход, при котором программа может выполнять несколько задач одновременно, не дожидаясь завершения каждой из них. Это позволяет повысить производительность программы и сделать её более отзывчивой. В Python асинхронное программирование реализуется с помощью модуля asyncio.

Модуль asyncio предоставляет набор инструментов для создания асинхронных программ. Он позволяет создавать задачи, которые могут выполняться параллельно, и управлять их выполнением. Асинхронные программы работают по принципу «неблокирующего ввода-вывода», что означает, что они не блокируют основной поток выполнения при ожидании ответа от сети или других ресурсов. Вместо этого они продолжают выполнение других задач, пока ждут ответа.

import asyncio

async def fetch\_url(url):

response = await asyncio.get\_sem\_awake(url)

return response.text()

def main():

loop = asyncio.get\_event\_loop()

tasks = [fetch\_url("https://www.google.com/") for \_ in range(5)]

results = loop.run\_until\_complete(asyncio.gather(\*tasks))

print(results)

if \_\_name\_\_ == "\_\_main\_\_":

main()

В этом примере мы создаём асинхронную функцию fetch\_url(), которая получает текст с указанного URL. Затем мы создаём список из пяти задач, каждая из которых вызывает функцию fetch\_url() с разным URL. Наконец, мы используем метод run\_until\_complete() для запуска всех задач и получения результатов. Метод gather() собирает результаты всех задач в один список.

**Области применения параллельного и асинхронного программирования**

Оба подхода широко используются в современных программных системах для повышения эффективности выполнения задач и оптимизации использования ресурсов. Однако они подходят для разных типов задач и условий работы, что делает их выбор важным аспектом разработки программного обеспечения.

Параллельное программирование подходит для задач, требующих интенсивных вычислений, таких как обработка изображений, научные расчёты или машинное обучение. Оно позволяет ускорить выполнение этих задач за счёт одновременной работы нескольких ядер процессора или нескольких машин в сети.

Пример использования параллельного программирования — задача обработки изображений. Для этого можно разделить изображение на несколько частей и обработать каждую часть на отдельном ядре процессора. Это позволит ускорить процесс обработки изображения по сравнению с последовательной обработкой.

Асинхронное программирование подходит для задач, связанных с вводом-выводом, таких как сетевые запросы, работа с базами данных или файловой системой. Оно позволяет избежать блокировки основного потока программы во время ожидания ответа от сервера или доступа к файлу. Это повышает отзывчивость программы и позволяет ей обрабатывать больше запросов одновременно.

Пример использования асинхронного программирования — веб-сервер, который обрабатывает запросы от клиентов. Для каждого запроса сервер может создать отдельный поток или процесс, чтобы не блокировать основной поток во время обработки запроса.

Это позволит серверу обрабатывать больше запросов одновременно и улучшить его производительность.

**Заключение**

Можно отметить, что параллельное и асинхронное программирование являются важными инструментами для повышения производительности и эффективности программ. Они позволяют ускорить выполнение задач, снизить нагрузку на серверы и улучшить взаимодействие с пользователем. Однако выбор конкретного подхода зависит от конкретных требований и условий задачи.

Также можно указать на перспективы развития параллельного и асинхронного программирования на Python. Например, можно ожидать появления новых инструментов и библиотек, которые упростят разработку параллельных и асинхронных приложений. Также возможно дальнейшее развитие технологий многопоточности и многозадачности, что позволит ещё больше повысить производительность программ.