**Лабораторная номер 6. Многопоточность в Python**

**Цель:** Изучение работы с параллельными вычислениями в языке Python

**Язык: Python.**

**Задача:** Реализовать алгоритм на Python, продемонстрировать увеличение производительности в за счет распараллеливания вычислений, сравнить работу с threading и multitasking (не надо дублировать код алгоритма вычислений, используйте код повторно).

**Немного теории:** В языке Python, а точнее в наиболее распространенном его интерпретаторе CPython, используется механизм глобальной блокировки (GIL) который существенно ограничивает скорость работы интерпретатора в многопоточном режиме. В каждый момент времени интерпретатор обрабатывает только один поток инструкций, что делает классические потоки крайне неэффективными в большинстве случаев.

Для работы с потоками в Python используются модули Threading и multitasking. Модуль threading обеспечивает работу с потоками внутри одного процесса, модуль multitasking обеспечивает распараллеливание вычислений между отдельными процессами, что позволяет обойти ограничение глобальной блокировки.

Интерфейсы модулей threading и multitasking практически совместимы, что значительно упрощает переход с одного на другой.

Учитывайте, что при использовании multiprocessing появятся существенные накладные расходы на передачу данных в дочерние процессы, следовательно, распараллеливание асимптотически простых алгоритмов вроде подсчета элементов или сортировки подсчетом будет неэффективно параллелиться на процессах. Высокая эффективность достигается в том случае, если входные-выходные данные алгоритма малы по сравнению с объемом вычислений (например, если передавать только имена файлов или параметры для генераторов данных)

**Индивидуальные задания.**

1. Сделать одно из заданий из предыдущих лабораторных.

2. Реализовать поиск текста в файлах (на входе искомый текст и путь к каталогу, содержащему множество текстовых файлов (не меньше 100), на выходе — список файлов и количество раз, сколько в каждом из них встречается искомая строка)

Усложненный вариант: программа должна продемонстрировать ускорение при параллельной обработке даже если файл в каталоге один. (распараллелить поиск текста в каждом файле)

3. Реализовать распараллеливание обработки изображений. Программа должна брать все файлы изображений из входного каталога и создавать в выходном их уменьшенные до заданного размера копии.

4. Распараллелить скачивание данных из интернета.

**Лабораторная номер 7.** Очереди задач.

**Задача:** Реализовать совместную обработку данных группой исполнителей.

Необходимо реализовать

- Приложение-сервер, обеспечивающее раздачу задач через очередь задач

- Приложение-исполнителя, получающее задачи от сервера и выполняющее их

Распределение задач должно осуществляться в пределах локальной сети (при тестировании можно показать работу на одном компьютере) или глобально, в зависимости от выбранной очереди. В качестве очереди задач можно использовать Celery, Amazon Simple Queue Service, Redis

**Варианты заданий:**

1. Скачивание информации из сети (Исполнителям передается список адресов, или диапазон, если адреса имеют какую то числовую природу, например, id страниц в социальной сети)

2. Перебор паролей к архиву (Исполнителям передается диапазон паролей и путь к файлу-архиву)

3. Что нибудь свое