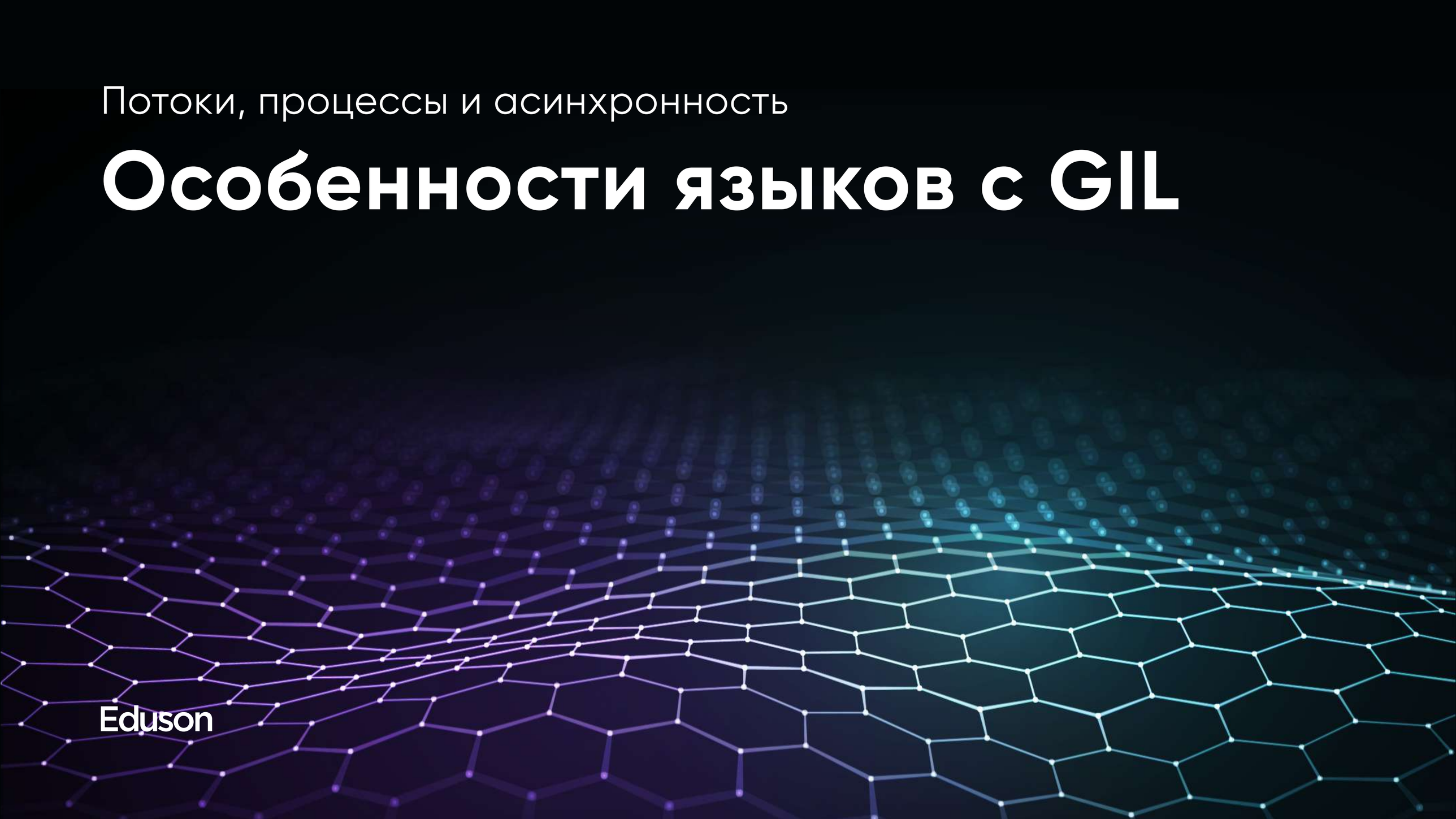
Потоки, процессы и асинхронность

Особенности языков с GIL



Eduson

Эксперт курса

Андрей Оськин

– backend developer, data engineer в TenTen (Япония);
– 5 лет в разработке на Python;
– product-manager и ментор в data science команде.

Eduson

Как работает GIL

Python Global Interpreter Lock (GIL) — это способ синхронизации потоков, который используется в некоторых интерпретируемых языках программирования, например, в Python и Ruby.

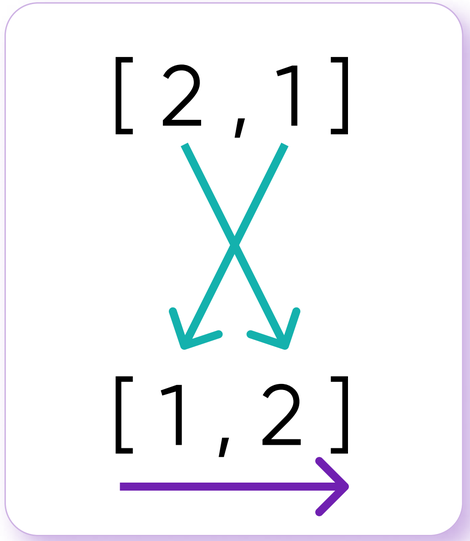
GIL — это система блокировки, которая позволяет только одному потоку управлять интерпретатором в каждый момент времени.

Пример непредвиденного результата в сортировке

Представьте, что одновременно с одним списком работают две программы:

1. Сортировка по возрастанию.
2. Считывание элементов по очереди.

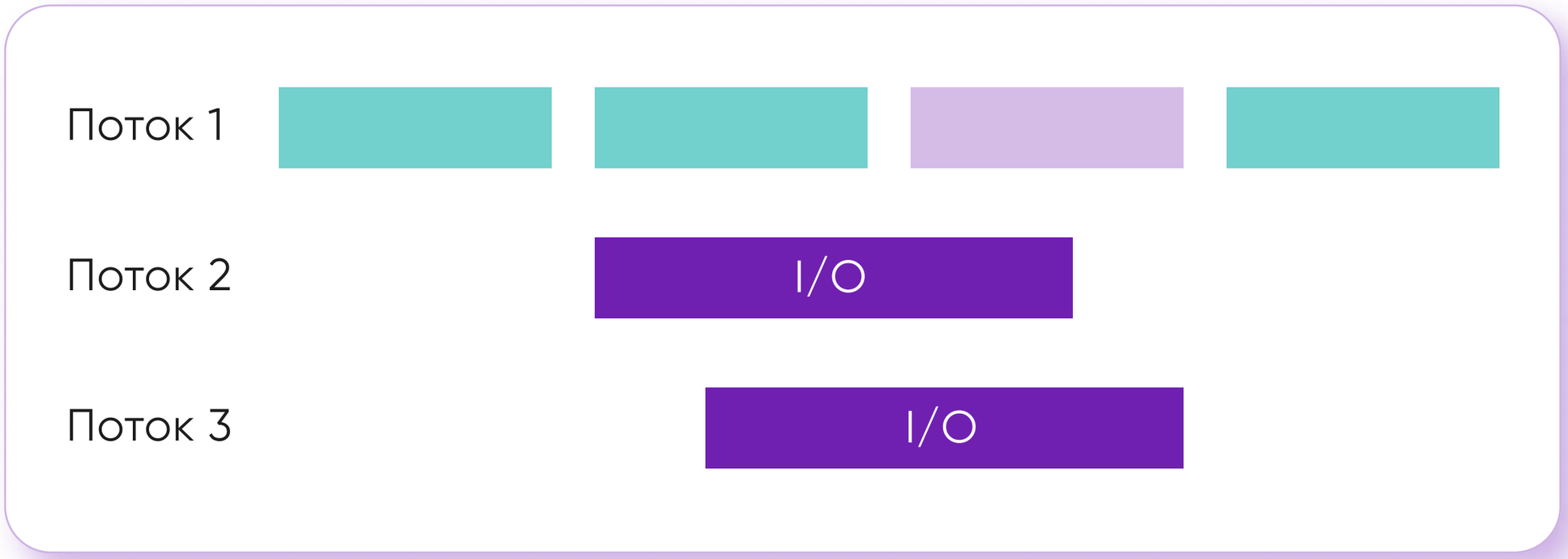
Обе программы работают над общим объектом - списком. И от очередности выполнения зависит результат. Например вторая программа может прочитать [2, 2].



Такой результат является непредсказуемым поэтому небезопасным. Значит операция не является Thread Safe.

Запросы ввода и вывода могут выполняться параллельно

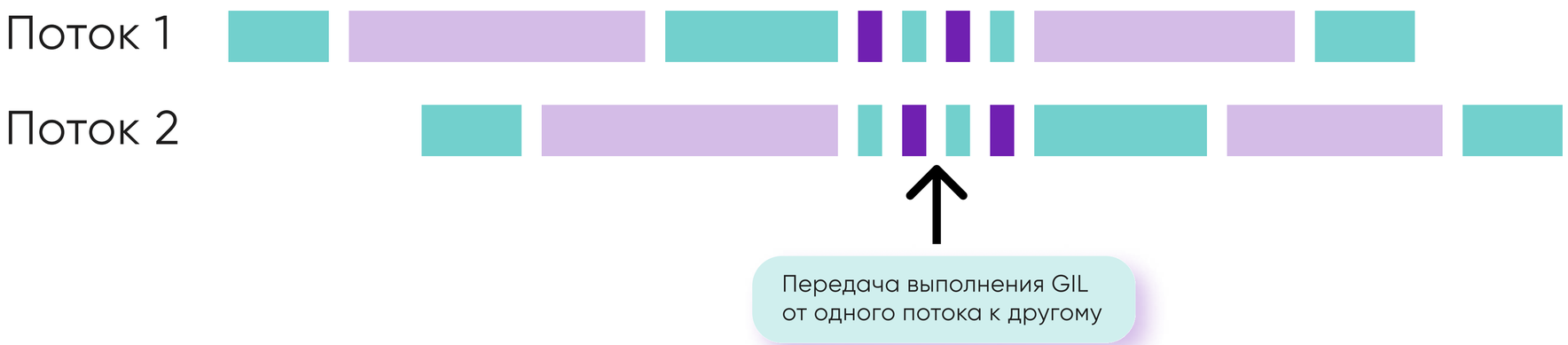
GIL допускает параллельность этого типа запросов, что ускоряет выполнение программы.



При использовании многопоточности нужно написать и запустить несколько копий в отдельных потоках. Это заполняет простои операций ввода и вывода (I/O).

Как работают две программы с I/O на разных потоках

Пример запуска двух копий такой программы:



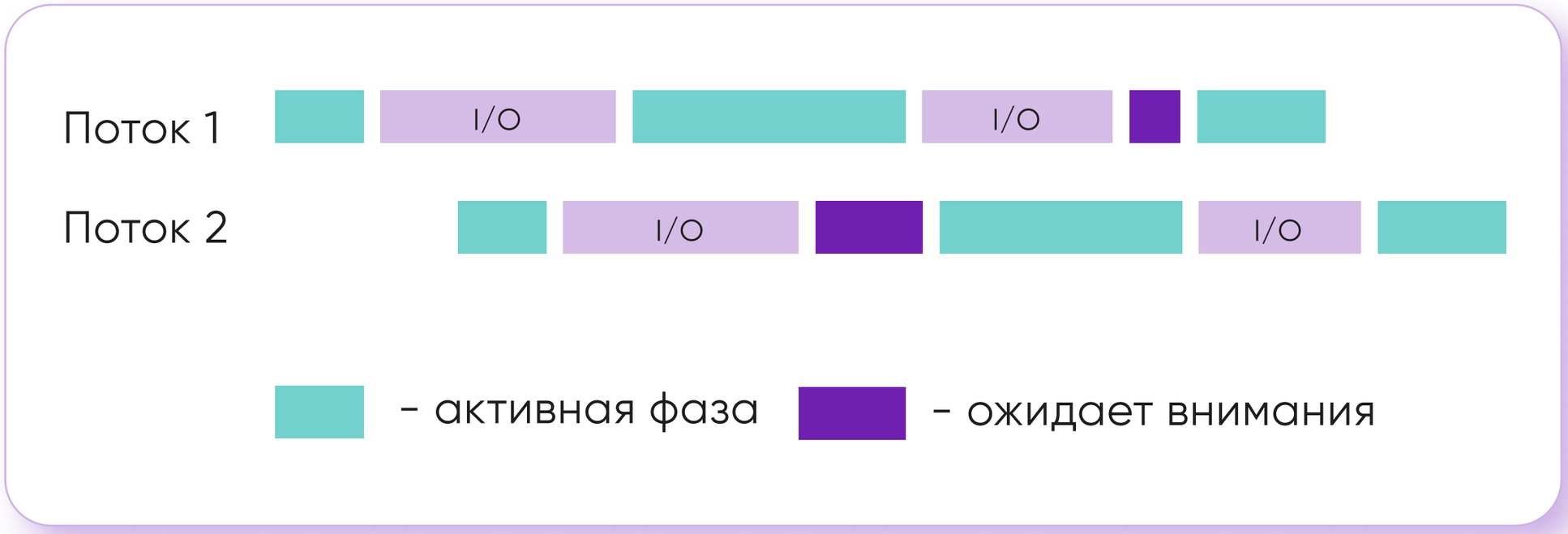
GIL помогает равномерно продвигать выполнение программ, когда их вычисления пересекаются.

Не нужно думать о времени отклика приложения, даже если одна программа захочет занять все время вычислениями.

Корутины упрощают ассинхронное выполнение

Корутины - это потоки исполнения кода, которые работают поверх системных потоков.

Асинхронная программа выполняется в одном потоке, но I/O операции не блокируют выполнение.



Этот подход решает проблему доступа к общим объектам, так как место переключения внимания выбрано программистом, а исполнение идет только в одном потоке.

Как обойти ограничения GIL при выполнении CPU-bound задач

CPU-Bound задачи - это задачи на вычисление, которые не связаны с I/O. Чтобы ускорить их выполнение, необходимо каждую запустить в отдельном процессе.

Запуск нового процесса — это запуск нового интерпретатора со своим GIL. Программа может ускориться и занять больше одного физического ядра.

При запуске нового процесса необходимо скопировать весь контекст программы. Это вызывает задержку. И иногда создание нового процесса занимает столько же времени, что и расчет.

Потоки легче всего писать

Применение: Количество параллельных потоков ~100

Сложности и ограничения:

- затрудняется получение и передача результата между потоками;
- параллельное исполнение возможно только для операций I/O;
- появляется нагрузка по переключению между потоками.

Примеры потоков: Django, Web server.

Корутины позволяют переключать потоки без дополнительной нагрузки

Применение: Большое количество одновременных клиентов

Сложности и ограничения:

- требуется опыт создания корутин для эффективной работы;
- нужны асинхронные библиотеки.

Примеры корутин: FastAPI, Async Web Server.

Процессы являются настоящим параллельным исполнением

Применение: эффективно работает только для распараллеливания длительных операций

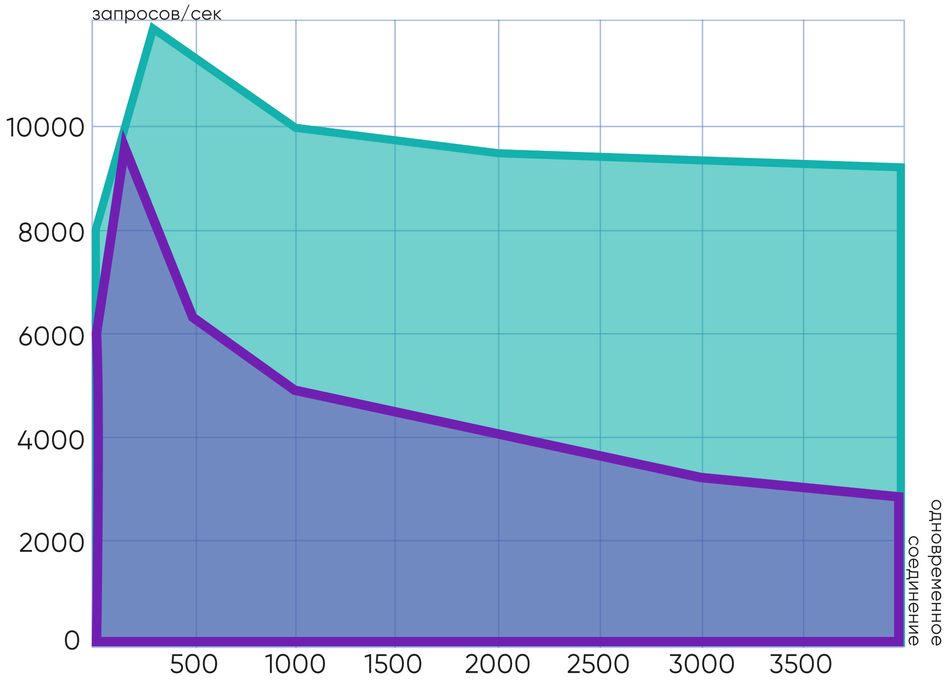
Сложности и ограничения:

- генерация процессов требует большой дополнительной нагрузки на сервер и железо.

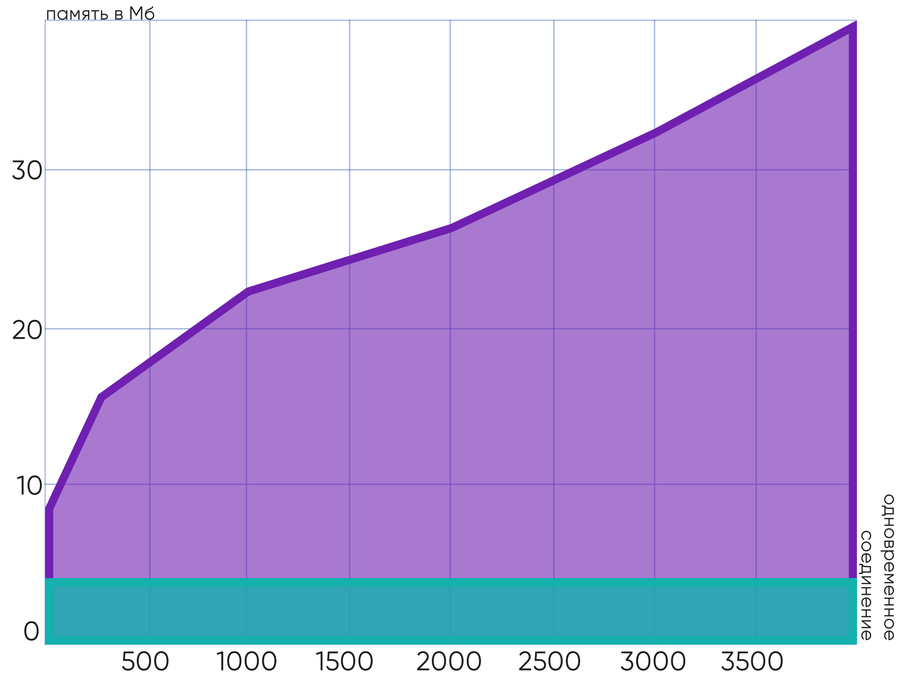
Примеры процессов: pandparallel, joblib.

Сравнение многопоточности Apache и ассинхронности NGINX

Отношение конкуренции к Запрос/сек



Отношение конкуренции к памяти



- Apache - NGINX

Выберите один подход для решения задачи

1. Мы хотим запустить прототип web-приложения. Пока не известны все требования к приложению.
2. Нам надо проверить один расчет со множеством различных входных данных.
3. Нужно ускорить один долгий расчет.
4. Хотим выделить одну часть нашего высоконагруженного веб приложения, чтобы масштабировать ее отдельно.
5. Мы хотим сделать парсер информации с web-страниц. Требуется быстрая обработка страницы. Сохраняем результат в БД.