

Параллельное и многопоточное программирование

Thread and Multiprocessors



Задача: Загрузка файлов

- Представьте, что вы создаёте программу, которая должна скачать 10 разных файла из интернета. Каждая загрузка занимает N минуты.

Проблема:

- Программа должна ждать **полные N секунды** для завершения загрузки Файла №1.
- Только после этого она начинает ждать Файл №2, затем Файл №3 и т.д.

Общее время: $(10 * N)$ секунд. Программа простояивает $10*N$ минут, просто **ожидая** данные по сети.

Решение

- Вы назначаете каждую из 10 загрузок в потоки.
- Вы запускаете все 10 потоков одновременно.
- Когда Поток 1 начинает загрузку Файла №1, он сразу же сталкивается с ожиданием ответа от сервера.
- Пока Поток 1 ждёт, Поток 2, 3, и все остальные 10 потоков начинают свои загрузки.

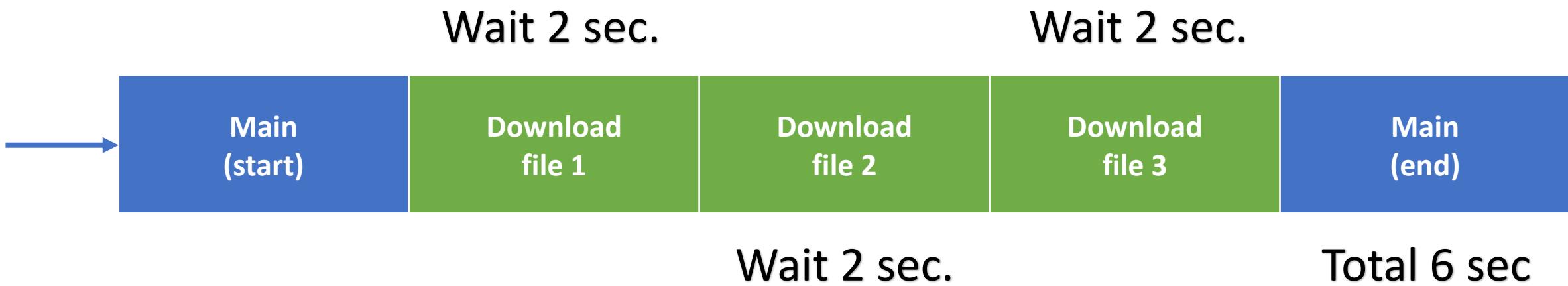
Многопоточность (Threading)

- Многопоточность в Python использует модуль `threading` и лучше всего подходит для задач, которые требуют много **ожидания** (I/O-bound).
- **Как это работает:** Все потоки используют одно и то же пространство памяти. Запускаются они быстро.
- **GIL (Глобальная Блокировка Интерпретатора)**

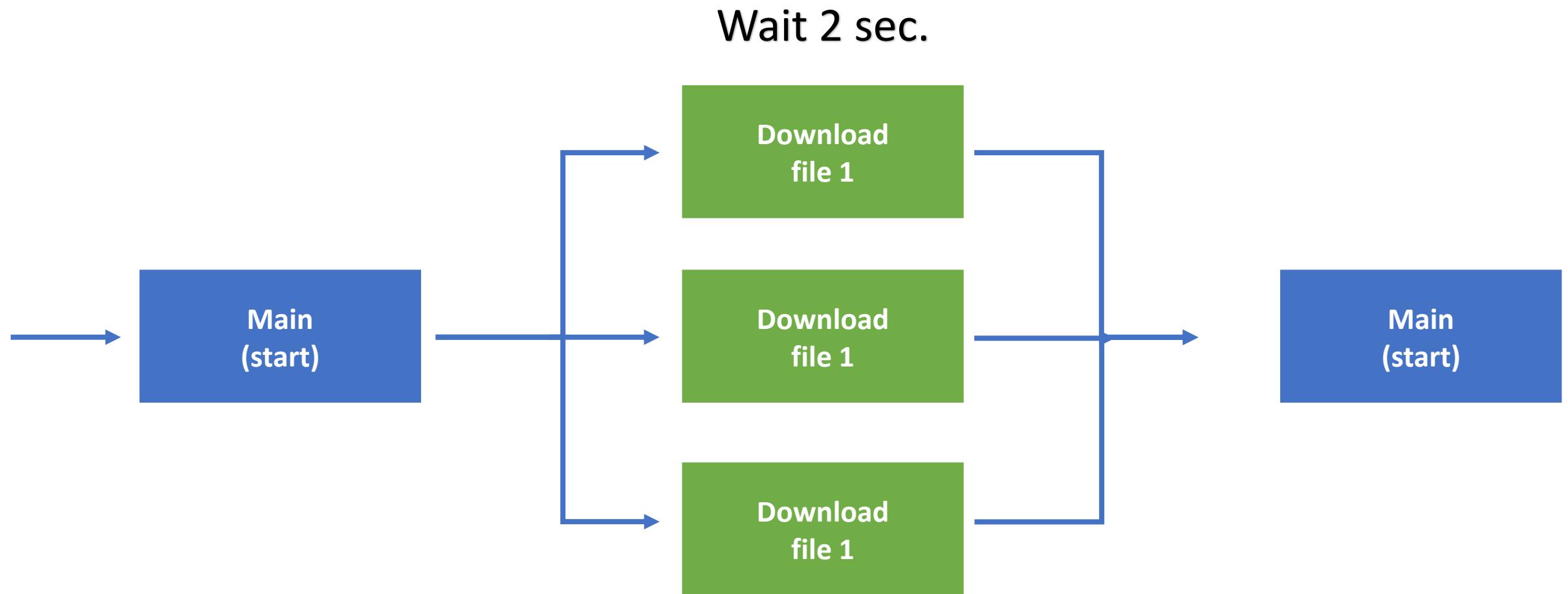
Правило Шефа: Одиночный шеф (GIL) может держать только один инструмент (запускать один поток) одновременно.

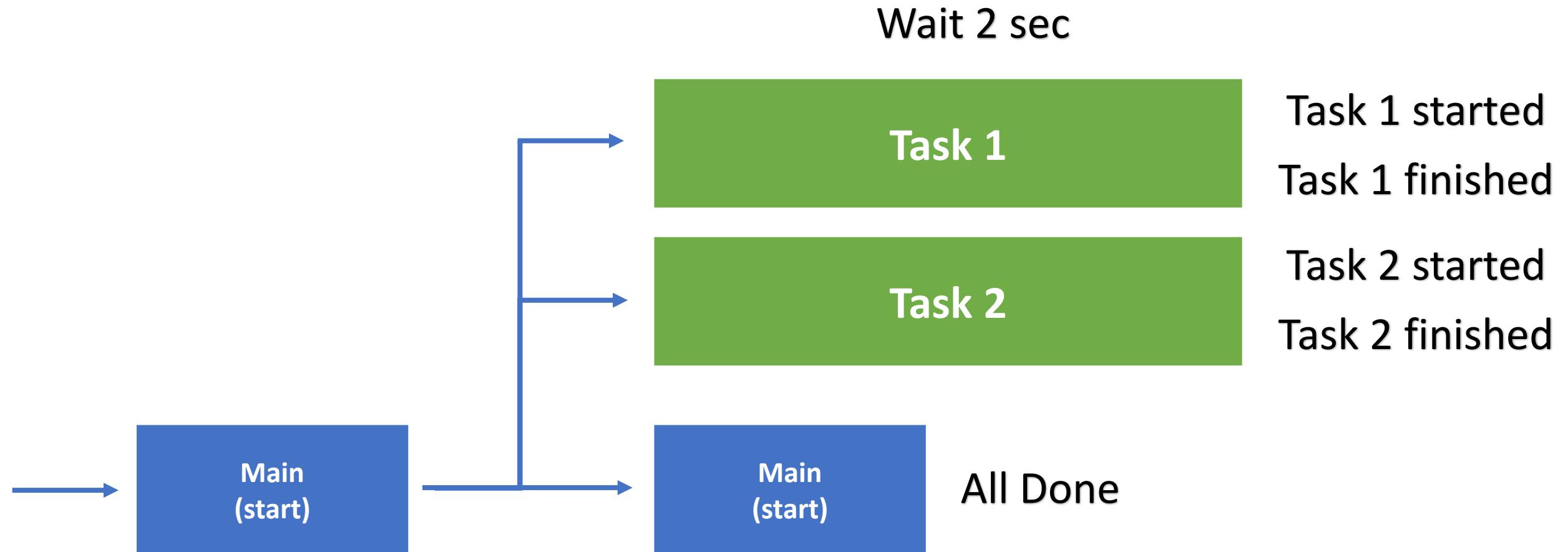
- **Когда это работает лучше всего (I/O-Bound)**

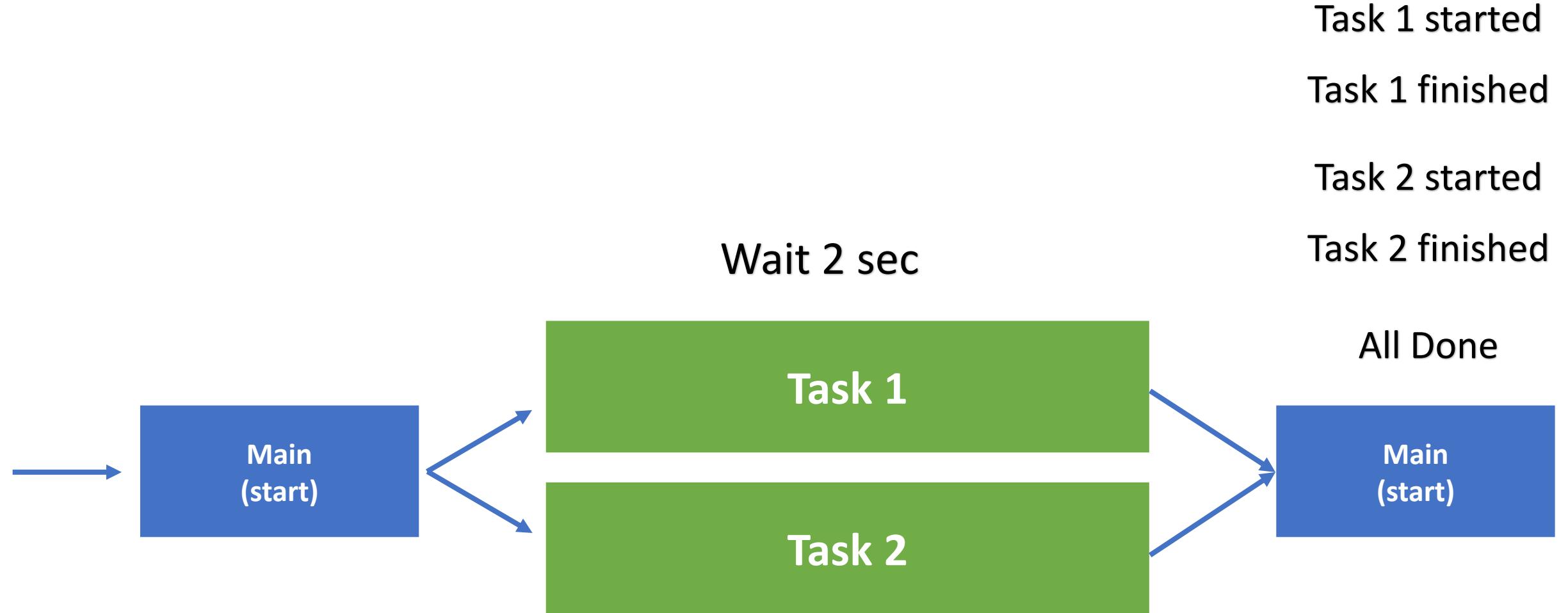
Загрузка файла (1 поток)

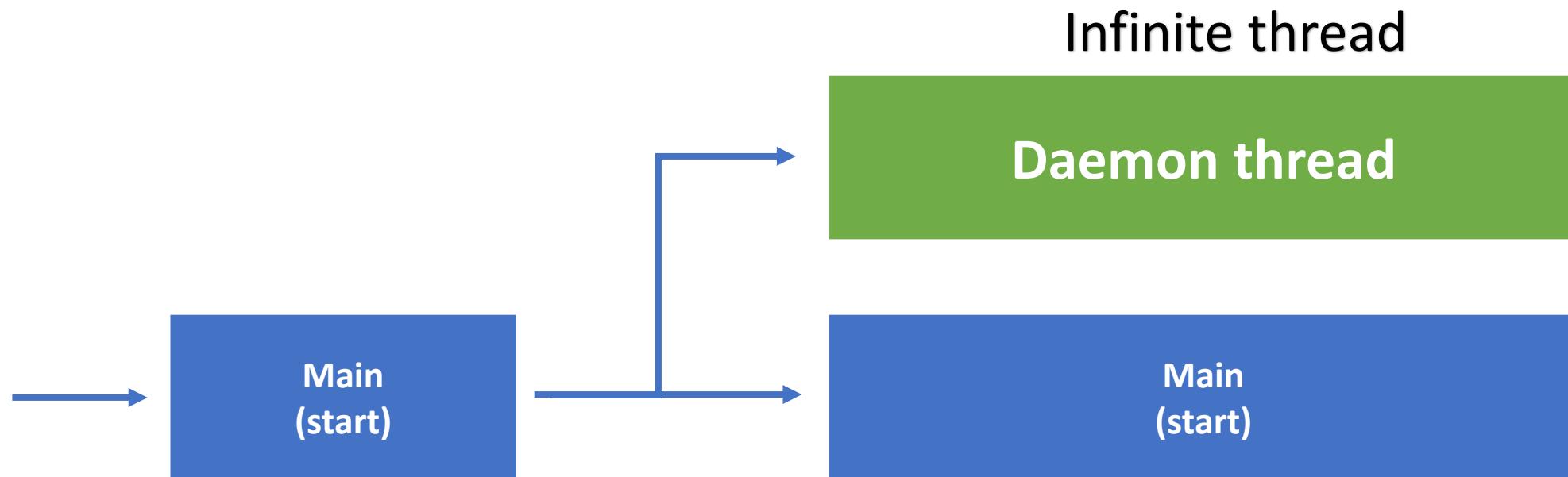


Загрузка файла (4 поток)









Конкурентность и Параллелизм

Определение Конкурентности (Concurrency): Задачи кажутся выполняющимися одновременно (чередующееся выполнение). Обеспечивает **отзывчивость** (скрытие задержек).

- *Аналогия:* Шеф-повар режет овощи, затем быстро помешивает кастрюлю, затем продолжает резать овощи — быстро переключается между задачами.

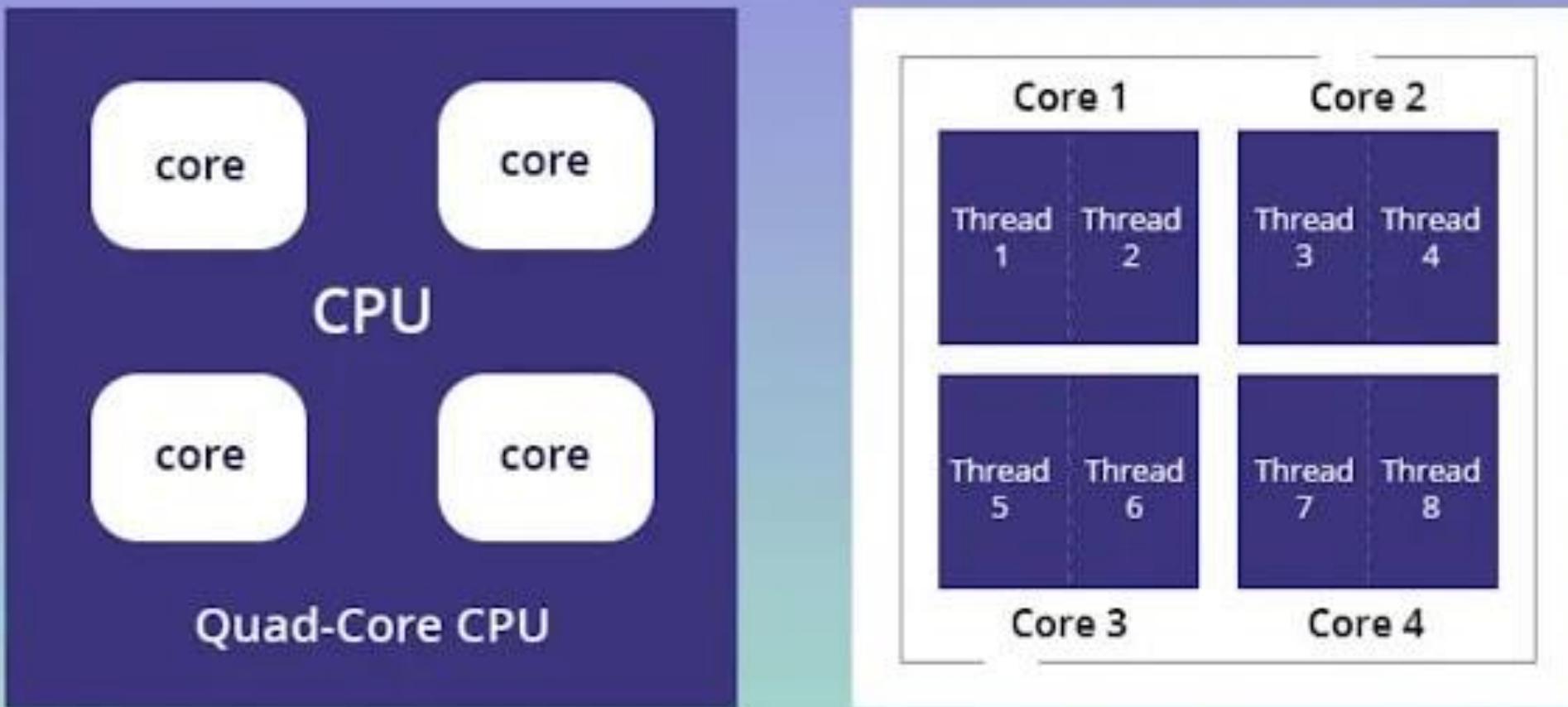
Конкурентность и Параллелизм

Определение Параллелизма (Parallelism): Задачи действительно выполняются одновременно на нескольких ядрах процессора. Обеспечивает **скорость** (сокращение общего времени выполнения).

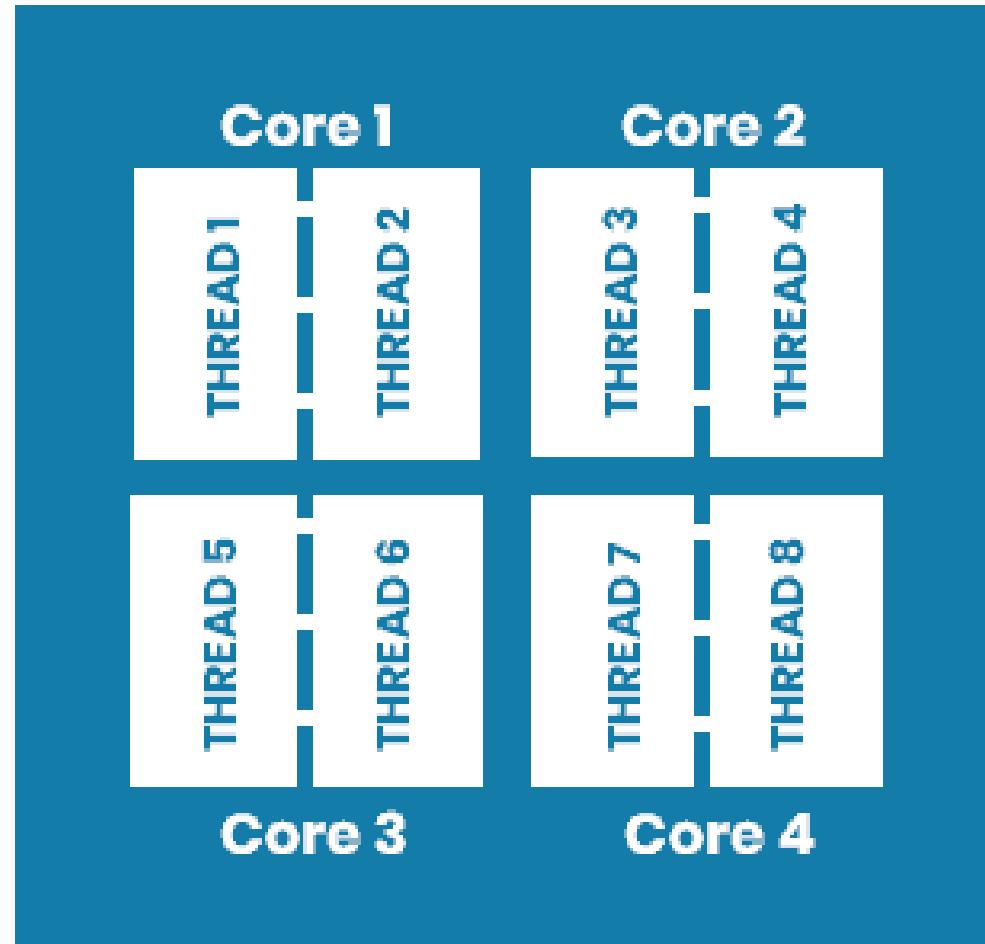
- *Аналогия:* Два разных повара (или помощника на кухне) работают над двумя отдельными задачами **одновременно**.

Цель: Оптимизировать производительность программы, используя время ожидания (I/O) или несколько ядер (CPU).

CPU Core vs Thread



CPU THREADS



Многопроцессность (Multiprocessing)

Многопроцессность использует модуль `multiprocessing` и является единственным способом достижения истинного параллелизма для сложных вычислений (CPU-bound).

Как это работает: Каждый новый процесс — это **полноценный, отдельный экземпляр Python** со своей собственной памятью. Обмениваться данными между ними сложнее.

Ключевое Преимущество: Поскольку каждый процесс отделен, **каждый получает свой собственный GIL (своего собственного шефа)**. Это позволяет коду Python выполнять одновременно на всех доступных ядрах процессора.

Многопроцессность (Multiprocessing)

- **Когда это работает лучше всего (CPU-Bound):** Когда задача требует интенсивных вычислений без ожидания, например, обработка больших массивов, рендеринг графики или сложное моделирование.

Случай использования: Анализ данных, кодирование видео, математические расчеты.