CPU bound операции в короутинах

Синхронные операции в короутинах исполнять бесполезно, потому что выполнение короутин станет последовательным.

```
import asyncio
import time

SUB_PROGS_COUNT = 100

async def worker(_id: int):
    i = 0
    begin = time.time()
    while i < 50_000:
        i += 1

    print(f"ID: {_id} FINISH duration: {time.time() - begin}")

async def main():
    begin = time.time()
    await asyncio.gather(
        *[worker(i) for i in range(SUB_PROGS_COUNT)]
    )
    print(f"ALL COROs DONE duration:{time.time() - begin}")

asyncio.run(main())
```

```
....
ID: 96 FINISH duration: 0.0011718273162841797
ID: 97 FINISH duration: 0.0011668205261230469
ID: 98 FINISH duration: 0.0011720657348632812
ID: 99 FINISH duration: 0.0011718273162841797
ALL COROs DONE duration:0.12114882469177246
```

Никакого выигрыша во времени мы не получили.

Такую же картину мы получим при вычислениях в разных тредах, т.к. обе сущности ограничены 1 процессом, и, следовательно, 1 ядром, его обрабатывающим.

В случае многопроцессной обработки наш выигрыш во временим был в N раз больше, где N — число ядер, задействованных в обработке процессов.