

Atividade 5

Otimizações em Python

Integrantes :

Felipe Seiji Momma Valente	12543700
Fernando Gonçalves Campos	12542352
Thiago Shimada	12691032

Descrição da Atividade

Para esta atividade, o grupo optou utilizar um código de monte carlo para se estimar o valor de π , a biblioteca utilizada para se verificar o tempo de execução foi o `timeit`. Já na otimização, o código já contava com o uso do `numpy`, mas adicionamos o `numba` para uma otimização mais efetiva.

`Numba` é uma biblioteca de aceleração para Python que usa compilação JIT (Just-In-Time) via LLVM para transformar funções numéricas puras de Python/NumPy em código de máquina otimizado. Para usar esta biblioteca, basta adicionar um decorador antes da função que deseja otimizar.

Em nosso código foi utilizado o decorador `@njit` (just in time nopython mode) para compilar a função sem a necessidade do interpretador Python, o que costuma gerar os maiores ganhos (frequentemente de 10 a 100 vezes em grandes loops). Além disso, foi adicionada a opção `cache = True`. Essa opção salva o código compilado, para que as próximas execuções do script não precisem compilar a função desejada novamente.

Dados Obtidos

Os dados foram obtidos executando cada iteração do código 100x e variando o valor de `n`, sendo `n` a quantidade de pontos gerados para se estimar o valor de π . Além disso, cada `n` foi executado utilizando o `numba/numpy` e somente utilizando o `numpy`.

n = 1.000:

- Tempo de execução Numpy : 0,7822998s
- Tempo de execução Numpy + Numba : 0,9749469s

n = 10.000:

- Tempo de execução Numpy : 7,7834776s
- Tempo de execução Numpy + Numba : 1,0755938s

n = 100.000:

- Tempo de execução Numpy : 84,7669334s
- Tempo de execução Numpy + Numba : 1,7272759s

n = 1.000.000:

- Tempo de execução Numpy : Não executado
- Tempo de execução Numpy + Numba : 5,9570952s

n = 10.000.000:

- Tempo de execução Numpy : Não executado
- Tempo de execução Numpy + Numba : 52,6055303s

Conclusão

A análise dos dados evidencia que a execução do código com somente o numpy fica até 50x mais lento do que se comparado com o código com o numba, entretanto com valores de n pequenos (como 1000), o overhead adicionado com todo o processo do numba não compensa o ganho de tempo na otimização que ele entrega, fazendo assim ele ficar mais lento do que o código sem ele. Além disso o numba possibilitou a execução de n maiores ainda que não seriam executados sem ele, executando um n 100x maior com um tempo de execução menor.