Acelerando Aplicações Científicas com Cython

Emanuel Lima

Motivação

Fibonacci Recursivo em C, Go e Python

Python

```python
def fib(num):
    if num == 1 or num == 2:
        return 1
    else:
        return fib(num -1) + fib(num - 2)

def main():
    print(fib(40))

if __name__ == "__main__":
    main()
```

- Levou 28,16s para executar.
- Python é uma linguagem dinamicamente tipada, interpretada e com garbage collector.

Go

```go
package main

import "fmt"

func main() {
    fmt.Println(fib(40))
}

func fib(num int32) int32 {
    if num == 1 || num == 2 {
        return 1
    }
    return fib(num-1) + fib(num-2)
}
```

- Levou 0,59s para executar.
- Go é uma linguagem estaticamente tipada, compilada e com garbage collector.

C

```c
#include <stdio.h>

long int fib(int num)
{
    if (num == 1 || num == 2)
        return 1;
    else
        return fib(num - 1) + fib(num - 2);
}

int main()
{
    long int n;
    n = fib(40);
    printf("%ld \n", n);
    return 0;
}

```

- Compilado com -02 levou 0,24s para executar.
- Sem a flag de otimização, 0,55s
- C é uma linguagem estaticamente tipada, compilada e sem garbage collector.

Como melhorar a performance do Python?

Usando Cython!

- É uma linguagem que engloba o Python, compilada, opcionalmente estaticamente tipada.
- Se desejado, também é possível gerenciar a memória manualmente.
- Um simples comando instala o compilador:
  `pip install cython`

Como Funciona?

- Processo manual:
- Um código Cython é salvo com a extensão .pyx
- Você deve compilá-lo para um código em C e depois compilar esse código C, criando, assim, um arquivo .so (Shared Object) que você importará no seu programa Python
- `cython arquivo.pyx` (Produz um arquivo.c)
- `cythonize -i arquivo.pyx` (Produz um arquivo.c e já o compila)
- Esse modo “manual” de fazer as coisas só é usado para testes
- O Cython não cria executáveis, apenas bibliotecas dinâmicas. (.so)
- Vamos ver como isso funciona na prática

Como Funciona?

- O modo usual de se utilizar o Cython é com a ferramenta `setup.py` do módulo Distutils
- `python setup.py build_ext --inplace`
- Vejamos na prática.

Sintaxe da Linguagem: Tipos Estáticos e Funções C

Sintaxe da Linguagem: “Classes” C

Usando Cython com Numpy

Referências

- D. Seljebotn, Fast numerical computations with Cython in Proceedings of the 8th Python in Science conference (SciPy 2009), G. Varoquaux, S. van der Walt, J. Millman (Eds.), pp. 15-22
- S. Behnel, R. Bradshaw, D. Seljebotn, Cython tutorial in Proceedings of the 8th Python in Science conference (SciPy 2009), G. Varoquaux, S. van der Walt, J. Millman (Eds.), pp. 4-14
- https://cython.readthedocs.io/en/latest/
- https://gitlab.com/emanuellima/cython-pybr