2019/20

Programação Paralela e Distribuída

Prática 5: Python multiprocessing

(https://docs.python.org/3/library/multiprocessing.html)

1 – Construa um programa para somar um array (numpy) de 10,000,000 de inteiros, entre -5 e +5.

O programa principal deve gerar os inteiros a somar, criar tantos processos quantos os "cores" do seu computador e dividir o trabalho em partes iguais pelos vários processos.

- O número de cores do seu computador deve ser determinado de forma automática. Use o package: *psutil* .

- Cada processo deve receber como parâmetro a secção do array com os valores a somar.

- Cada processo ao terminar a sua soma parcial deve enviar o resultado ao processo principal através de um pipe.

2 – Pretende-se um programa em crie vários processos e cada um desses processos irá escrever um ficheiro com um conjunto de inteiros. O processo P1 irá gerar o ficheiro f1 com os valores de 0 a 999, o processo P2 irá gerar o ficheiro f2 com os valores de 1000 a 1999, isto é, o processo P*i* irá gerar o ficheiro f*i* com os valores entre (i-1)*1000 a (i*1000)-1 .

O número de processos a usar deverá ser 2 * o número de cores do seu computador. O programa principal deverá avisar o utilizador qual o tempo necessário total de execução, e qual o número de ficheiros gerados.

3 – Construa um programa que crie um conjunto de processos em número igual a 2 * o número de cores do seu computador. Cada um desses processos deve ler um dos ficheiros gerados no exercício 2. O processo Pi deve ler o ficheiro fi.

- Para cada valor do ficheiro deve calcular a função fx (ver figura 1), e calcular o somatório dos valores obtidos. No final cada processo deve enviar essa soma ao processo principal através de um pipe.

- O processo principal deve mostrar o somatório das somas parciais.

def fx (x):
if (x % 2 == 0):
  r = math.sin(x)
else:
  r = math.cos(x)
return r
	<- Figura 1

4 – Modifique agora o exercício 3 de forma que:

- Após o momento em que todos os processos já leram os ficheiros de dados, o programa principal elimine esses ficheiros da sua diretoria. Use um objeto do tipo Barrier para saber que já todos os processos leram os ficheiros de dados. Cada processo deverá prosseguir como no exercício 3, calculando o valor de fx para cada valor lido, calcular o somatório e enviar ao processo principal.

Atenção: não remova ficheiros os ficheiro errados.

5 – Suponha agora que o seu programa principal vai gerar um array com inteiros entre 0 e (2*numCores * 1000 -1) sendo numCores o número de cores do seu computador.

Use uma Pool de processos (com número de processos igual a numCores) e a instrução *map* para calcular a função fx em cada um dos valores gerados acima.

O programa principal deverá depois calcular o somatório dos valores obtidos com a execução do map na pool de processos.

6 – Modifique o exercício anterior tal que em vez da instrução map, use a instrução não bloqueante, map_async. O programa principal depois de invocar o map_async na pool de processos deverá pedir ao utilizador o nome de um ficheiro para onde deverá escrever o resultado obtido da pool de processos, quando este estiver disponível.

7 – Construir um programa em que o processo principal gera uma matriz quadrada (1000 x 1000). Queremos construir uma nova matriz em que cada valor é substituído pelo valor do somatório dos valores que o cercam na matriz (ver figura da página que se segue). Para isso o programa deve receber do utilizador o número de processos p que quer usar para fazer o trabalho. Iremos dividir a matriz em p em blocos (de 1000/p linhas)

Cada processo irá construir um bloco da matriz resultado e devolver ao programa principal. A comunicação deve ser feita por pipes.