112-PROBA-pi-paralelo

April 23, 2018

Cálculo aproximado de π contando cuantos puntos aleatorios del cuadrado $[0,1] \times [0,1]$, de entre un total de N, caen dentro de la circunferencia de radio unidad, digamos que han caido dentro N_0. El área de un cuarto de circunferencia, $\pi/4$ es aproximadamente igual al cociente N_0/N:

```
In [1]: def pi(N,rs):
            cont = 0
            set_random_seed(rs) #Inicializamos el generador de numeros aleatorios
            for muda in xsrange(N):
                x,y = random(),random() #Coordenadas de un punto aleatorio del cuadrado
                if x^2+y^2 <= 1:
                    cont += 1
            return cont
```

Usando un único núcleo calcula la aproximación de π usando $4 * 10^7$ puntos aleatorios en unos 90 segundos, y el valor de π obtenido tiene 3 cifras decimales correctas.

```
In [2]: time pi(4*10^7,9873223)
```

```
CPU times: user 1min 10s, sys: 1.24 s, total: 1min 12s
Wall time: 1min 10s
```

```
Out[2]: 31418544
```

```
In [3]: ((4*31418544)/4*10^7).n()
```

```
Out[3]: 3.14185440000000e14
```

Usando los cuatro núcleos (dos núcleos reales y cada uno de ellos 2 virtuales), mandamos a cada núcleo 10^7 puntos y obtenemos un tiempo de 54 segundos.

```
In [4]: @parallel(4)
        def pi_paralelo(N,rs):
            cont = 0
            set_random_seed(rs)
            for muda in xsrange(N):
                x,y = random(),random()
                if x^2+y^2 <= 1:
                    cont += 1
            return cont
```

```
In [5]: time list(pi_paralelo([(10^7,23459),(10^7,29954),(10^7,7654),(10^7,67543)]))

CPU times: user 4 ms, sys: 16 ms, total: 20 ms
Wall time: 18.1 s


Out[5]: [(((10000000, 7654), {}), 7854508),
         (((10000000, 67543), {}), 7855063),
         (((10000000, 29954), {}), 7854510),
         (((10000000, 23459), {}), 7853909)]

In [6]: pi1 = 4*(7854510+7854508+7855063+7853909)/(4*10^7);(pi1).n()

Out[6]: 3.14179900000000
```

Usando dos núcleos el tiempo total es 58 s, y vemos que la ventaja obtenida por usar los núcleos virtuales es mínima.

```
In [7]: @parallel(2)
        def pi_paralelo(N,rs):
            cont = 0
            set_random_seed(rs)
            for muda in xsrange(N):
                x,y = random(),random()
                if x^2+y^2 <= 1:
                    cont += 1
            return cont

In [8]: time list(pi_paralelo([(2*10^7,23459),(2*10^7,29954)]))

CPU times: user 0 ns, sys: 12 ms, total: 12 ms
Wall time: 36.2 s


Out[8]: [(((20000000, 23459), {}), 15707559), (((20000000, 29954), {}), 15706709)]
```

£Mejora la aproximación de π al aumentar N (el número de puntos aleatorios que consideramos)?

```
In [9]: [pi(10^k,965452) for k in srange(3,9)]

Out[9]: [789, 7833, 78468, 785370, 7855040, 78536148]
```

£Qué opinas?