Implementación de funciones con Threads (documento echo en LaTeX)

Yefer Andersson Mamani Chambi
Universidad Nacional del Altiplano
yefer.andersson.unai@gmail.com
Puno

17 de Octubre de 2022

1 Funcion numPar con Threads

1.1 Código en python

[67]:
```
from threading import Thread
import time
import math
def numPar1(n):
    time_ini = time.time()
    print("***Inicio numPar1***\n")
    i = 2
    while(i <= n):
        print(i)
        i = i + 2
    print("Total: ", math.floor(n/2))
    print("***Fin numPar1***\n")
    time_end = time.time()
    total = time_end - time_ini
    print("Tiempo: ", total)

def numPar2(n):
    time_ini = time.time()
    print("***Inicio numPar2***\n")
    for i in range(1,n+1):
        if i % 2 == 0:
            print(i)
    print("Total: ", math.floor(n/2))
    print("***Fin numPar2***\n")
    time_end = time.time()
    total = time_end - time_ini
    print("Tiempo: ", total)
```

1.2 Para n = 10

[68]:
```
t1 = Thread(target=numPar1, args=(10,))
t2 = Thread(target=numPar2, args=(10,))
t1.start()
t2.start()
```

```
***Inicio numPar1***
2
***Inicio numPar2***
4
2
6
4
8
10
6
Total:  5
8
***Fin numPar1***
10
Tiempo:  0.0019948482513427734
Total:  5
***Fin numPar2***
Tiempo:  0.0019948482513427734
```

1.3 Para n = 30

[69]:
```
t1 = Thread(target=numPar1, args=(30,))
t2 = Thread(target=numPar2, args=(30,))
t1.start()
t2.start()
```

```
***Inicio numPar1***
2
***Inicio numPar2***
4
2
6
4
8
6
10
8
10
12
12
14
14
```

```
16
16
18
18
20
20
22
24
26
22
28
24
30
26
Total:  15
***Fin numPar2***
28
Tiempo:  0.003989696502685547
30
Total:  15
***Fin numPar1***
Tiempo:  0.004987239837646484
```

1.4 Para n = 200

[70]:
```
t1 = Thread(target=numPar1, args=(200,))
t2 = Thread(target=numPar2, args=(200,))
t1.start()
t2.start()
```

```
***Inicio numPar1***
***Inicio numPar2***
2
2
4
6
4
8
6
10
8
12
10
14
12
16
14
18
```

```
16
20
18
22
20

*El resto de números aqui en medio*
*lo borre para no ocupar espacio en el documento*

192
190
194
192
196
194
198
196
200
198
Total:  100
200
***Fin numPar2***
Total:  100
Tiempo:  0.020658016204833984
***Fin numPar1***
Tiempo:  0.02166438102722168
```

1.5 Para n = 1000

[71]:
```
t1 = Thread(target=numPar1, args=(1000,))
t2 = Thread(target=numPar2, args=(1000,))
t1.start()
t2.start()
```

```
***Inicio numPar1***
***Inicio numPar2***
2
2
4
4
6
8
6
10
8
12
10
14
```

```
12
16
14
18
16
20
18
22
20
24
22
26
28
24
30
32
34
26
36
38
40
42
28
44
30
32
34

*EL resto de numeros aqui en medio*
*Lo borre para que no ocupe mucho documento*

996
954
998
956
1000
958
Total:  500
960
***Fin numPar2***
962
Tiempo:  0.15259265899658203
964
966
968
970
972
974
```

```
976
978
980
982
984
986
988
990
992
994
996
998
1000
Total:  500
***Fin numPar1***
Tiempo:  0.1545860767364502
```

[ ]:

[ ]:

2 Interpretación de resultados

Los resultados muestran que aunque la funnción 'numPar1' parece ser la mas sencilla, al ejecutarlos al mismo tiempo usando hilos, se muestra una mejor eficiencia con la función 'numPar2', ya que es la que acaba primero y en menor tiempo