MULTIPROCESSING vs ESECUZIONE SEQUENZIALE

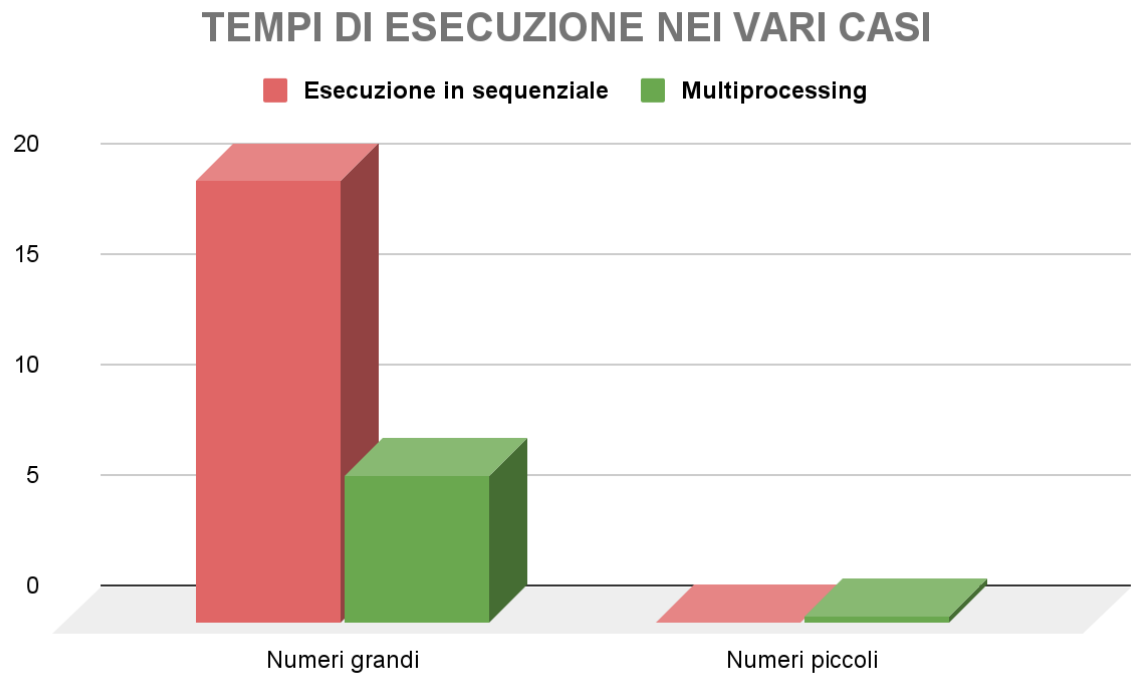
Per effettuare il confronto tra l'esecuzione in sequenziale e quella con l'utilizzo del multiprocessing ho utilizzato il seguente codice dove ho fatto due esempi sul calcolo del fattoriale: uno con numeri molto grandi e l'altro esempio con l'utilizzo di numeri molto piccoli.

```python
import time
import multiprocessing as mp

def fattoriale(n):
        f=1
        for i in range (1, n+1):
            f*=i
        return f

def main():
    numeri = 100000, 100001, 100002, 100003

    #loop for
    st = time.time()
    risultati = []
    for i in numeri:
        risultati.append(fattoriale(i))
    ft = time.time()
    print("Tempo utilizzando l'esecuzione in sequenziale di numeri grandi:", ft-st)

    #multiprocessing
    st = time.time()
    with mp.Pool(processes=6) as pool:
        result = pool.map(fattoriale, numeri)
    ft = time.time()
    print("Tempo utilizzando il multiprocessing di numeri grandi:", ft-st)

    #-------------------------------------------------------

    numeri = 500, 501, 502, 503

    #loop for
    st = time.time()
    risultati = []
    for i in numeri:
        risultati.append(fattoriale(i))
    ft = time.time()
    print("Tempo utilizzando l'esecuzione in sequenziale di numeri piccoli:", ft-st)

    #multiprocessing
    st = time.time()
    with mp.Pool() as pool:
        result = pool.map(fattoriale, numeri)
    ft = time.time()
    print("Tempo utilizzando il multiprocessing di numeri piccoli:", ft-st)

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

Il risultato dei quattro calcoli approssimativamente è il seguente:

```
Tempo utilizzando l'esecuzione in sequenziale di numeri grandi: 20.735378980636597
Tempo utilizzando il multiprocessing di numeri grandi: 6.69771146774292
Tempo utilizzando l'esecuzione in sequenziale di numeri piccoli: 0.0009472370147705078
Tempo utilizzando il multiprocessing di numeri piccoli: 0.31116795539855957
```

Nel primo segmento di codice infatti, dove ho utilizzato il ciclo for per calcolare quattro numeri di grandi dimensioni, il programma ha impiegato, a completare le sue istruzioni, un tempo di circa 21 secondi. Nel secondo segmento invece, dove ho utilizzato il multiprocessing per calcolare quattro numeri di grandi dimensioni, il programma ha impiegato un totale di 6.7 secondi, un tempo molto buono. Al contrario con il primo codice, il secondo con numeri molto più piccoli ha dei risultati completamente stravolti, dove infatti nel primo segmento si risulta un tempo molto inferiore rispetto al secondo segmento, dove ho utilizzato il multiprocessing, con un tempo che si può approssimare agli 0 secondi.



Da questi risultati si può dedurre che:

Per calcoli più pesanti per la CPU è consigliato utilizzare il multiprocessing in quanto, come mostrato nel grafico, si presenta una notevole riduzione dei tempi di esecuzione. Al contrario, per quanto riguarda calcoli leggeri, è più consigliato, rispetto al multiprocessing, l'esecuzione sequenziale oppure l'utilizzo del multi-threading, poiché il tempo di esecuzione è leggermente inferiore.