

Carmelo Santamaria 514926
Federico Sciuto 517075

Genoma umano

La ricerca delle sequenze



Cos'è il genoma?

Il genoma umano è costituito da una lunga sequenza di DNA che contiene tutte le informazioni genetiche di un essere umano. In totale vi sono circa 3 miliardi di coppie di basi azotate, contenenti il codice genetico che determina le caratteristiche ereditate da una persona, come l'aspetto fisico, la predisposizione a determinate malattie e molto altro.

La mappatura del genoma umano è stata ultimata solo di recente (Marzo 2022) dopo più di 20 anni di ricerca, e ha un impatto significativo sulla medicina e la biologia, consentendo di studiare e comprendere meglio le basi genetiche di molte malattie e di sviluppare terapie mirate.



Sequenze rilevanti nel genoma umano

- **TATA:** nota come TATA-box è un elemento di controllo nella regione promotrice di molti geni, che interagisce con proteine per iniziare la trascrizione;
- **ATG:** sequenza di inizio della trascrizione (start codon);
- **TAA, TAG e TGA:** sequenze di terminazione della trascrizione (stop codon);
- **AAAAAA:** sequenza di poliadenilazione (poly-A tail), coinvolta nella stabilità e nel trasporto dell'mRNA;
- **TTAGGG:** ripetizioni telomeriche; sequenze che si trovano alle estremità dei cromosomi e proteggono i telomeri, contribuendo a preservare la stabilità dei cromosomi



Esecuzione del codice: scelta del file

```
carmelo@carmelo-GL65-Leopard-10SER: ~/Scrivania/UNIME/...  
carmelo@carmelo-GL65-Leopard-10SER:~/Scrivania/UNIME/ProLabReSiD23/PROgetto_funzionante$ python3 RicercaStringa_Multi.py  
Seleziona il file da analizzare:  
1 -> GRCh37_42MB_protein.faa  
2 -> GRCh38_105MB_protein.faa  
3 -> GRCh37_302MB_rna.fna  
4 -> GRCh38_742MB_rna.fna  
█
```

```
# Scelta del file da analizzare  
genome_files = input("\nSeleziona il file da analizzare:\n1 ->  
GRCh37_42MB_protein.faa\n2 -> GRCh38_105MB_protein.faa\n3 ->  
GRCh37_302MB_rna.fna\n4 -> GRCh38_742MB_rna.fna\n")  
  
file_mapping = {  
    "1": "GRCh37_42MB_protein.faa",  
    "2": "GRCh38_105MB_protein.faa",  
    "3": "GRCh37_302MB_rna.fna",  
    "4": "GRCh38_742MB_rna.fna"  
}  
  
filename = file_mapping.get(genome_files)  
  
if not filename:  
    print("\nOpzione non valida\n")
```

Una volta eseguito il codice, verrà data la possibilità di scegliere in quale file si desidera effettuare la ricerca. I primi due contengono sequenze di proteine, i successivi sequenze di RNA



Esecuzione del codice: inserimento stringa

```
carmelo@carmelo-GL65-Leopard-10SER: ~/Scrivania/UNIME/...
carmelo@carmelo-GL65-Leopard-10SER:~/Scrivania/UNIME/ProLabReSiD23/PROgetto_funz
ionante$ python3 RicercaStringa_Multi.py

Seleziona il file da analizzare:
1 -> GRCh37_42MB_protein.faa
2 -> GRCh38_105MB_protein.faa
3 -> GRCh37_302MB_rna.fna
4 -> GRCh38_742MB_rna.fna
1

Stringa da ricercare:
```

*I file **GRCh38** sono un'evoluzione dei **GRCh37**, contengono maggiori informazioni ed analisi più precise*

Dopo aver scelto il file, va inserita la stringa che si vuole ricercare all'interno di esso



Esecuzione del codice: scelta numero di thread

```
carmelo@carmelo-GL65-Leopard-10SER: ~/Scrivania/UNIME/...
carmelo@carmelo-GL65-Leopard-10SER:~/Scrivania/UNIME/ProLabReSiD23/PROgetto_funzionante$ python3 RicercaStringa_Multi.py

Seleziona il file da analizzare:
1 -> GRCh37_42MB_protein.faa
2 -> GRCh38_105MB_protein.faa
3 -> GRCh37_302MB_rna.fna
4 -> GRCh38_742MB_rna.fna
1

Stringa da ricercare: ACGT

Numero di processi da utilizzare (max = 12):
```

```
num_process_max = multiprocessing.cpu_count() # Numero di
core disponibili
num_process = int(input(f"\nNumero di processi da utilizzare
(max = {num_process_max}): "))

if num_process < 1 or num_process > num_process_max:
    print(f"\nNumero di processi inserito non valido
(deve essere compreso tra 1 e {num_process_max})\n")
```

Nei codici "Multithread" e "Ray", verrà anche richiesto di inserire il numero di thread da utilizzare per l'esecuzione



Esecuzione del codice: output

```
carmelo@carmelo-GL65-Leopard-10SER: ~/Scrivania/UNIME/...  
1 -> GRCh37_42MB_protein.faa  
2 -> GRCh38_105MB_protein.faa  
3 -> GRCh37_302MB_rna.fna  
4 -> GRCh38_742MB_rna.fna  
1  
  
Stringa da ricercare: ACGT  
  
Numero di processi da utilizzare (max = 12): 12  
  
La stringa 'ACGT' appare 166 volte nel file.  
Tempo di esecuzione: 6.184 secondi.
```

```
print(f"\nLa stringa '{stringa_target}' appare {conteggio_totale}  
volte nel file.")  
print(f"Tempo di esecuzione: {end_time - start_time:0.3f}  
secondi.\n")
```

Al termine dell'esecuzione verrà mostrato il conteggio delle occorrenze della stringa ricercata nel file, e il tempo impiegato espresso in secondi



Esecuzione di Ray

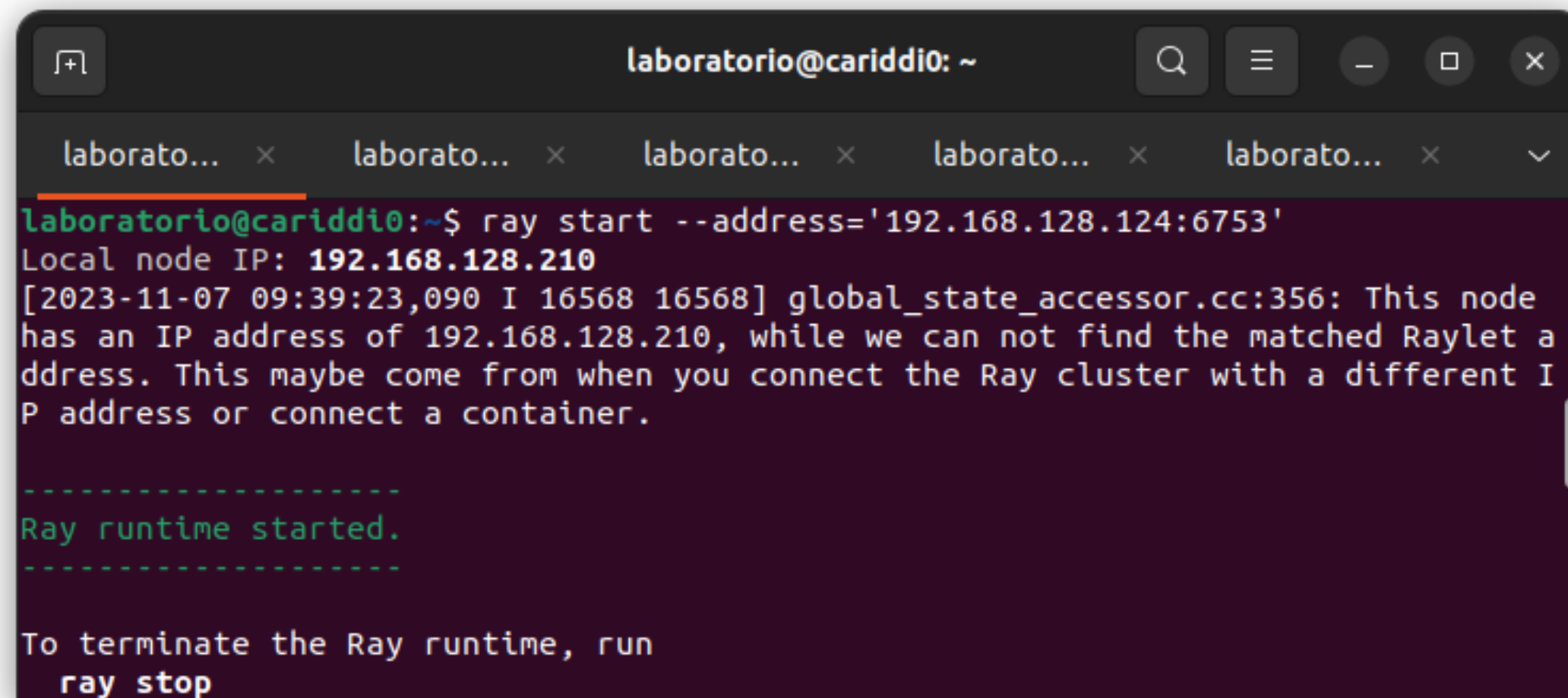
```
carmelo@carmelo-GL65-Leopard-10SER: ~  
carmelo@carmelo-GL65-Leopard-10SER:~$ ray start --head --port=6753  
Usage stats collection is enabled. To disable this, add '--disable-usage-stats'  
to the command that starts the cluster, or run the following command: 'ray disab  
le-usage-stats' before starting the cluster. See https://docs.ray.io/en/master/c  
luster/usage-stats.html for more details.  
  
Local node IP: 192.168.128.124  
  
-----  
Ray runtime started.  
-----  
  
Next steps  
To connect to this Ray runtime from another node, run  
  ray start --address='192.168.128.124:6753'  
  
Alternatively, use the following Python code:  
import ray  
ray.init(address='auto')  
  
To see the status of the cluster, use  
  ray status  
To monitor and debug Ray, view the dashboard at  
  127.0.0.1:8265  
  
If connection fails, check your firewall settings and network configuration.  
  
To terminate the Ray runtime, run  
  ray stop
```

Per poter creare un cluster sfruttando il framework “Ray”, abbiamo fatto partire la connessione tramite un nodo head (ovvero il nodo principale) con il comando:

ray start --head --port = “numero porta”



Esecuzione di Ray: start

A terminal window titled 'laboratorio@cariddi0: ~' with search, menu, and window control icons. It shows the execution of the 'ray start' command with the address '192.168.128.124:6753'. The output indicates the local node IP is 192.168.128.210 and that the Ray runtime has started. A warning message states that the node's IP does not match the Raylet address, suggesting a connection issue. Instructions for terminating the runtime are also provided.

```
laboratorio@cariddi0:~$ ray start --address='192.168.128.124:6753'
Local node IP: 192.168.128.210
[2023-11-07 09:39:23,090 I 16568 16568] global_state_accessor.cc:356: This node
has an IP address of 192.168.128.210, while we can not find the matched Raylet a
ddress. This maybe come from when you connect the Ray cluster with a different I
P address or connect a container.

-----
Ray runtime started.
-----

To terminate the Ray runtime, run
ray stop
```

Per collegare i nodi secondari (worker) al cluster, abbiamo utilizzato il comando:

ray start --address = "IP : porta"



Esecuzione di Ray: status

```
carmelo@carmelo-GL65-Leopard-10SER: ~  
carmelo@carmelo-GL65-Leopard-10SER:~$ ray status  
===== Autoscaler status: 2023-11-07 09:39:29.671700 =====  
Node status  
-----  
Healthy:  
1 node_d3d055c1698c4f469af08fb6fd178a5b1b0b7c04d53102fb5a01b6a1  
1 node_2dd5f5ed56e2096a0fbd7c384744e5f0a88ae10a4f9f9098f7c97ab8  
1 node_3762768e848cc15fa5e8c8c1b49f064937ca26c5f6443ec7ae998008  
1 node_7e6eaa737299c47b74a5a07e795be66ab9d0c96089f8393bc4881d59  
1 node_4e987dfdba03ae97d7bff1b5ef5fee5435610126eefcbed0d2848f03  
1 node_6ca752cafc57143d02192042f67ece0f6d6d6e67aa3933dcac215d2e  
Pending:  
(no pending nodes)  
Recent failures:  
(no failures)  
  
Resources  
-----  
Usage:  
0.0/32.0 CPU  
0.00/19.687 GiB memory  
0.00/9.049 GiB object_store_memory  
  
Demands:  
(no resource demands)
```

Per verificare

- il numero di nodi connessi;
- l'hardware (numero di thread e RAM) a disposizione

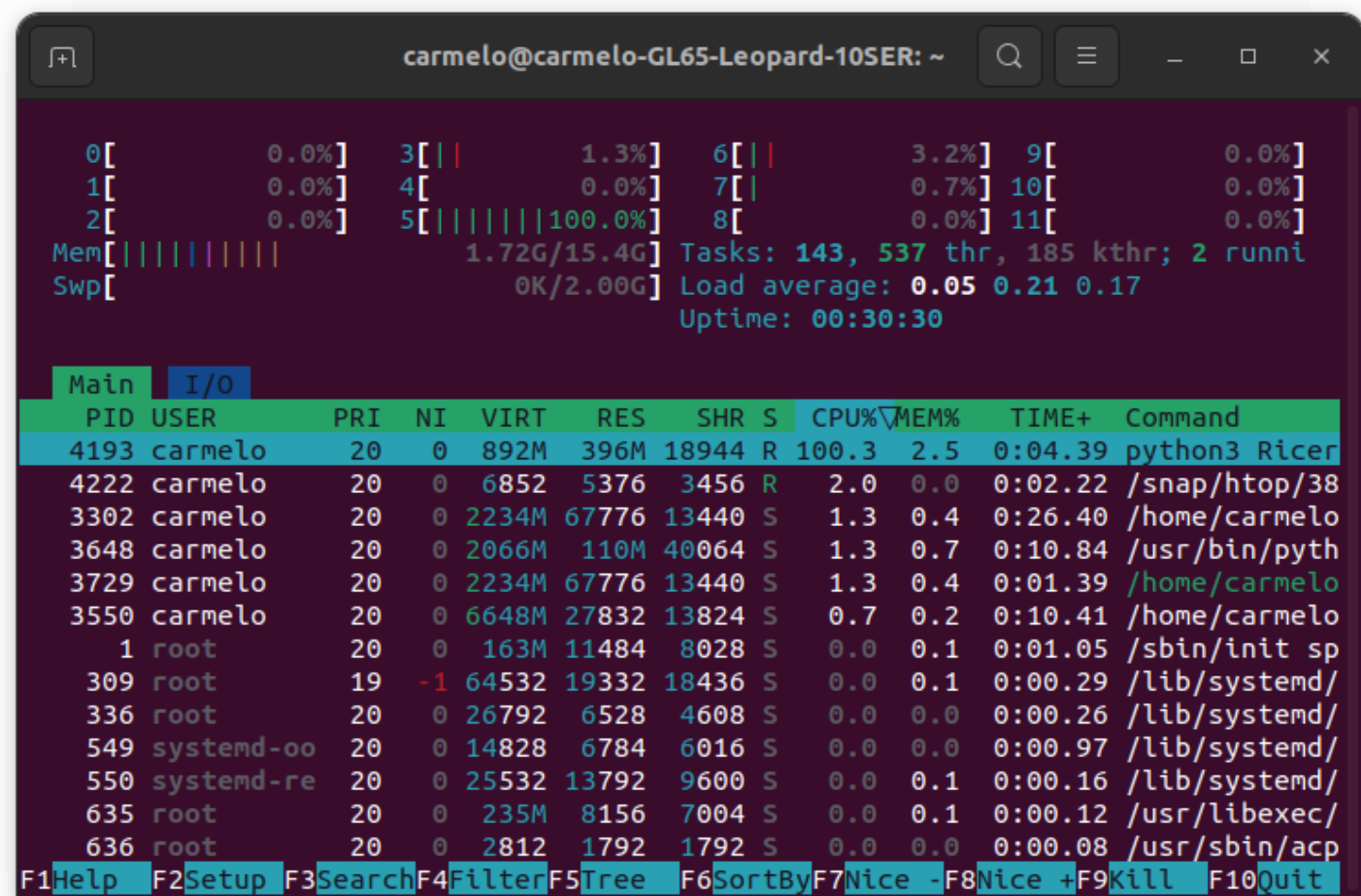
abbiamo utilizzato il comando:

ray status

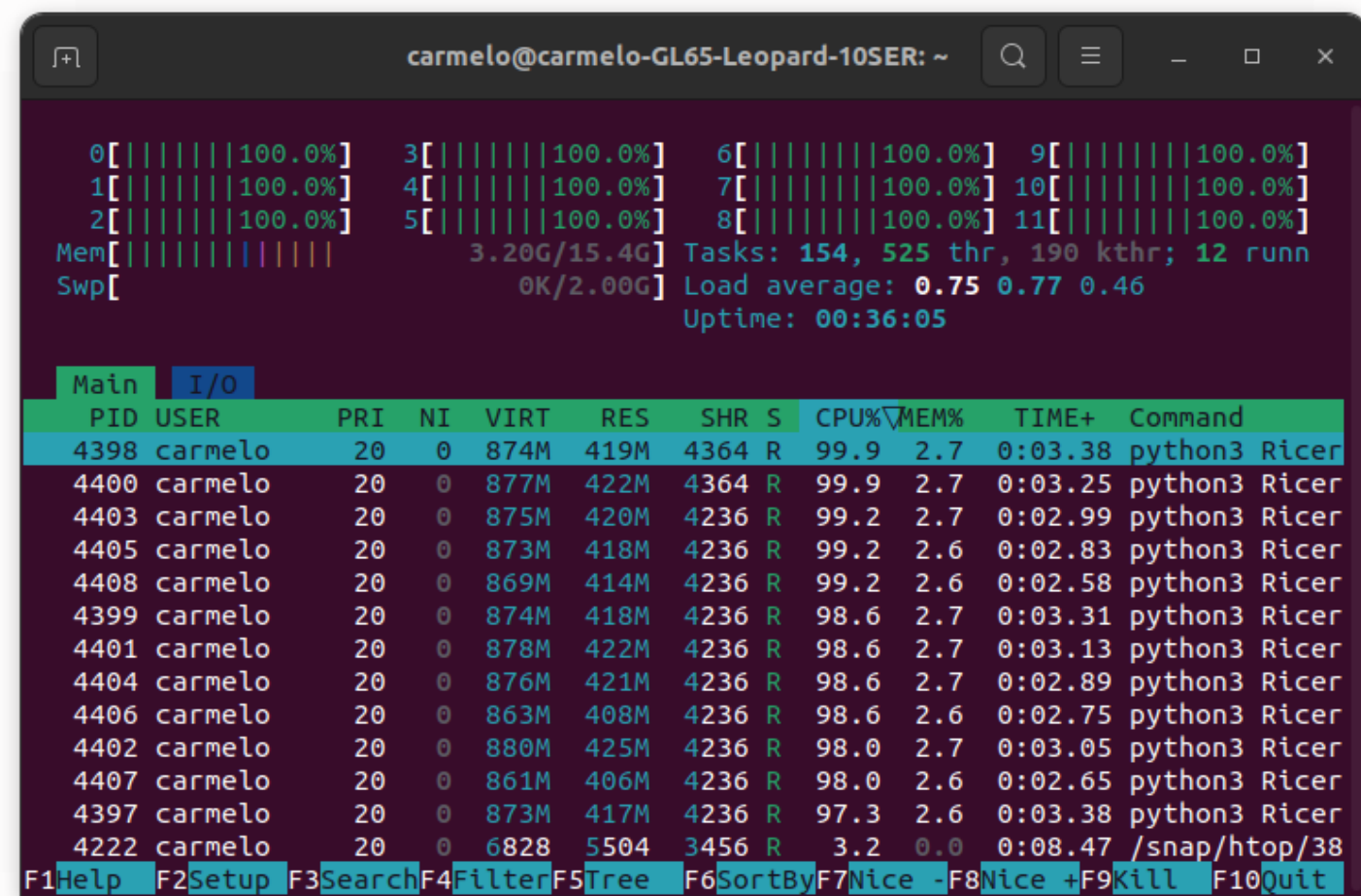


Carico di lavoro

Per controllare il carico di lavoro delle macchine in uso, abbiamo utilizzato Htop, un monitor di sistema interattivo per la visualizzazione e la gestione dei processi, che fornisce informazioni sulla CPU e sulla memoria RAM.



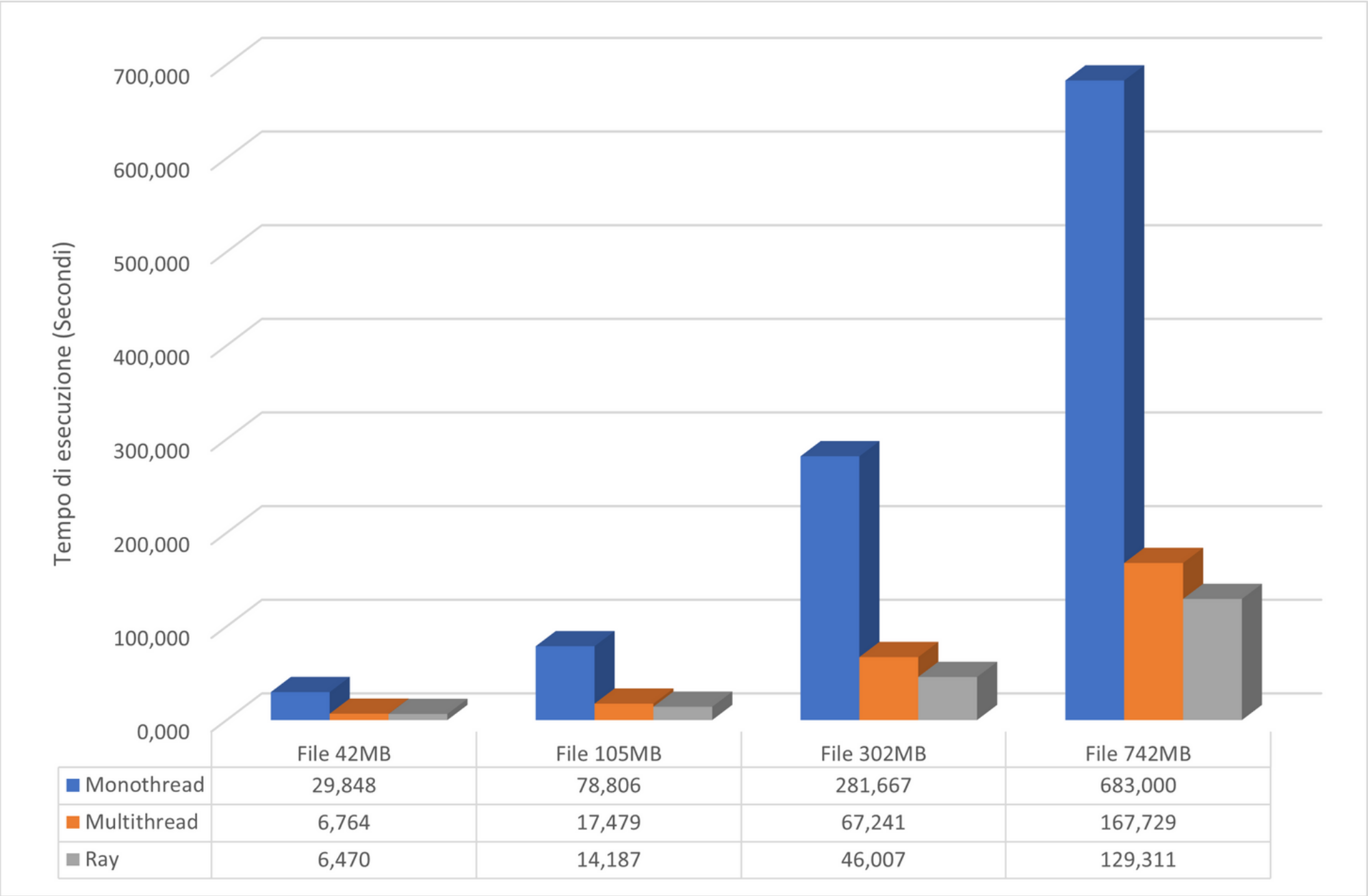
Esecuzione monothread



Esecuzione multithread



Risultati



In questo grafico sono rappresentati i tempi medi di esecuzione dei codici (*Monothread, Multithread, Ray*), per ciascun file (*42MB e 105MB proteine; 302MB e 742MB RNA*)