UNIVERSITÀ DEGLI STUDI FIRENZE

Faculty of Engineering

Master course in Artificial Intelligence

PERFORMANCE ANALYSIS REPORT FOR PASSWORD DECRYPTION

Parallel Programming for Machine Learning

Final-term assignment

Candidate
Loric TONGO GUIMTSA

Professor
Marco BERTINI
Associate Professor

Academic Year: 2023/2024

Contents

1. Introduction 3
2. Context 3
3. Methodology 3
4. Sequential vs Parallel 3
   4.1. Sequential Program 3
   4.2. Parallel Program 4
5. Results 5
   5.1. Sequential Program 5
      5.1.1. Execution time for: 5
   5.2. Parallel Program 5
      5.2.1. Execution time for: 5
6. Performance Analysis 6
7. Conclusion 7
8. Recommendations 7

1. Introduction

This report presents a performance analysis of sequential and parallel programs for decrypting passwords encrypted using the DES (Data Encryption Standard) algorithm. The sequential program decrypts passwords one by one, while the parallel program utilizes multiprocessing to decrypt passwords in parallel. The goal is to evaluate the speedup achieved by parallelizing the decryption process.

2. Context

The task involves decrypting passwords encrypted using the DES algorithm. Each password is 8 characters long and belongs to the character set [a-zA-Z0-9./]. The sequential program employs a simple approach to decrypt each password sequentially. In contrast, the parallel program utilizes multiprocessing to distribute the decryption tasks among multiple processes, aiming to accelerate the overall decryption process.

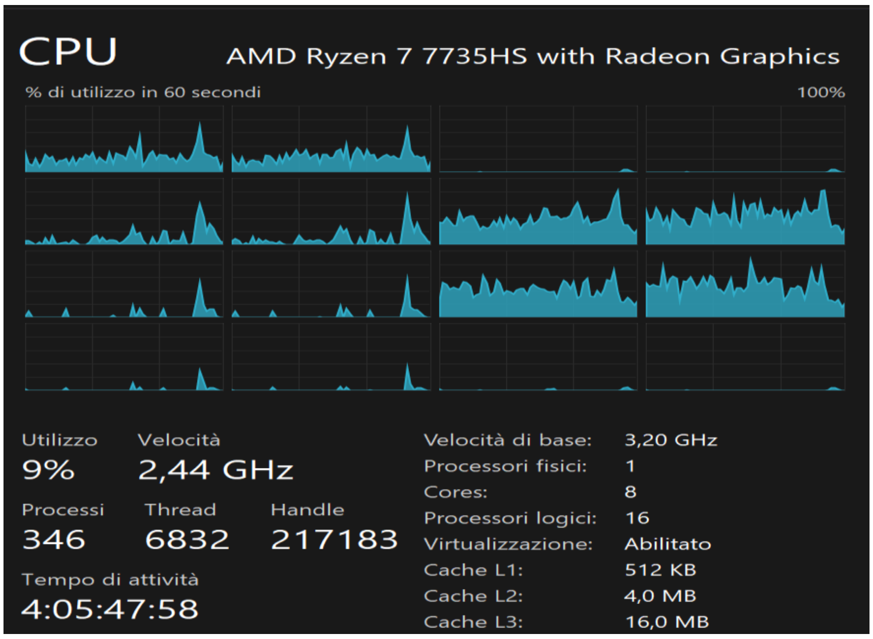
3. Methodology

Two programs were implemented for password decryption: a sequential program and a parallel program using multiprocessing. Both programs generate random passwords, encrypt them using DES, and then decrypt them. The performance of each program was measured in terms of execution time for encryption and decryption.

4. Sequential vs Parallel

4.1. Sequential Program

The sequential program generates random passwords, encrypts them using DES, and then decrypts them sequentially. It utilizes the "Crypto.Cipher.DES" module for encryption and decryption operations.


CPU
AMD Ryzen 7 7735HS with Radeon Graphics
% di utilizzo in 60 secondi
100%

Utilizzo 9%
Velocità 2,44 GHz
Processi 346
Thread 6832
Handle 217183
Tempo di attività 4:05:47:58

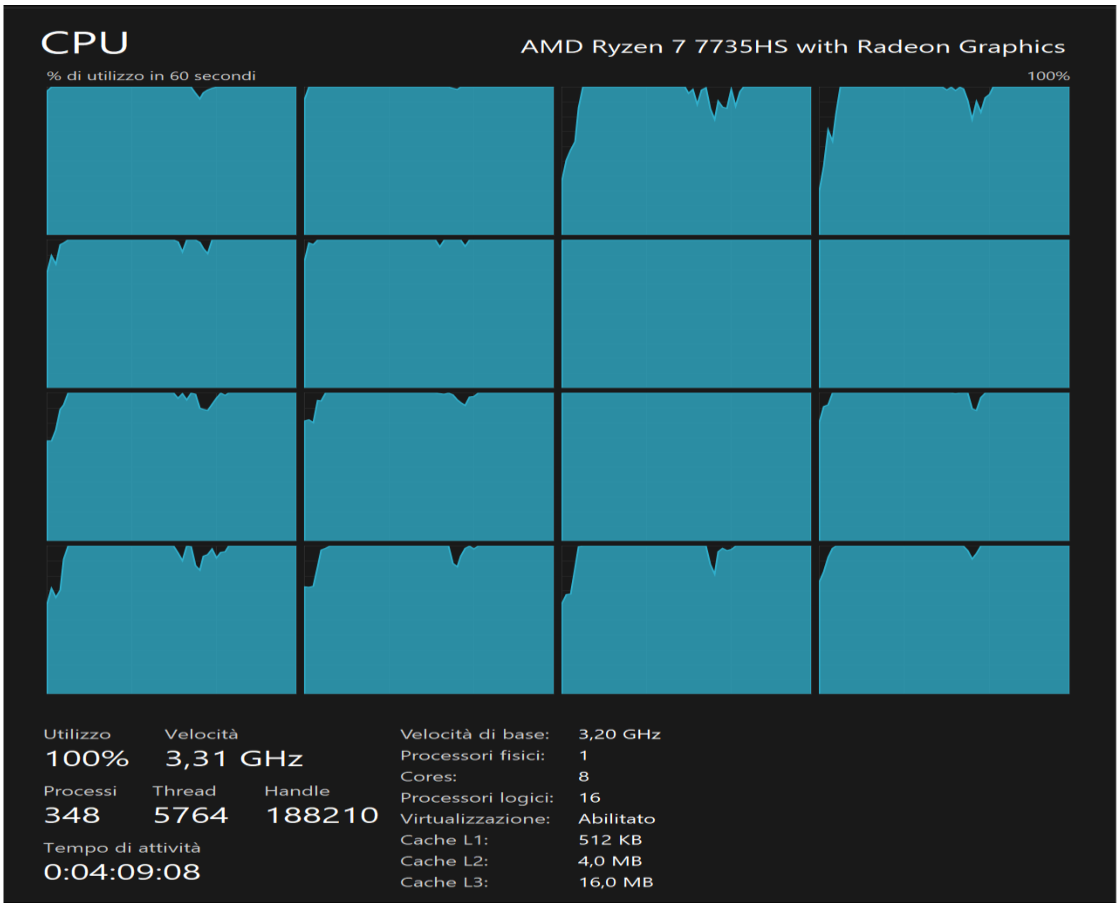
Velocità di base:	3,20 GHz
Processori fisici:	1
Cores:	8
Processori logici:	16
Virtualizzazione:	Abilitato
Cache L1:	512 KB
Cache L2:	4,0 MB
Cache L3:	16,0 MB

Figure 1: CPU utilization

As illustrated in Figure 1, which depicts CPU utilization during the execution of the sequential program, it's evident that the sequential program does not fully utilize all CPU cores. The figure clearly shows that only about 9% of the CPU power is utilized by the sequential program.

4.2. Parallel Program

The parallel program parallelizes the decryption process using multiprocessing. It divides the decryption tasks among multiple worker processes, each responsible for decrypting a subset of passwords. The "multiprocessing.Pool" class is used to manage the worker processes.


CPU
AMD Ryzen 7 7735HS with Radeon Graphics
% di utilizzo in 60 secondi
100%

```
Utilizzo      Velocità          Velocità di base:     3,20 GHz
100%          3,31 GHz          Processori fisici:    1
                                Cores:                8
Processi   Thread   Handle      Processori logici:    16
348        5764     188210      Virtualizzazione:     Abilitato
Tempo di attività               Cache L1:             512 KB
0:04:09:08                      Cache L2:             4,0 MB
                                Cache L3:             16,0 MB
```

Figure 2: CPU utilization

As evidenced in Figure 2, which depicts CPU utilization during the execution of the parallel program, it is evident that the parallel program utilizes all CPU cores. The figure clearly shows that the parallel program efficiently utilizes the entire processing power of the CPU.

5. Results

To compute the speedup, we will use this formulation:
Speedup = Sequential Execution Time / Parallel Execution Time

5.1. Sequential Program

5.1.1. Execution time for:

- Thousand (1000) passwords: 0.01 second*
- Ten thousand (10.000) passwords: 0.17 second*
- One hundred thousand (100.000) passwords: 1 second*
- One million (1.000.000) passwords: 14 seconds*
- Ten million (10.000.000) of passwords: 132 seconds*
- One hundred million (100.000.000) of passwords: 953 seconds*

5.2. Parallel Program

5.2.1. Execution time for:

- Thousand (1.000) passwords:
  - Using one (1) thread:
    - Execution time: 0.1 second*
    - **Speedup** = 0.01 / 0.1 = 0.1
  - Using sixteen (16) threads:
    - Execution time: 0.8 second*
    - **Speedup** = 0.01 / 0.8 = 0.01

- Ten thousand (10.000) passwords:
  - Using one (1) thread:
    - Execution time: 0.4 second*
    - **Speedup** = 0.17 / 0.4 = 0.4
  - Using sixteen (16) threads:
    - Execution time: 1 second*
    - **Speedup** = 0.17 / 1 = 0.17

- One hundred thousand (100.000) passwords:
  - Using one (1) thread:
    - Execution time: 1.7 second*
    - **Speedup** = 1 / 1.7 = 0.5
  - Using between 2 – 16 threads:
    - Execution time: 1 second*
    - **Speedup** = 1 / 1 = 1

- One million (1.000.000) passwords: 1.7 second*
  - Using one (1) thread:
    - Execution time: 14.7 seconds*
    - **Speedup** = 14 / 14.7 = 0.9
  - Using five (5) threads:
    - Execution time: 4 seconds*
    - **Speedup =** 14 / 4 = 2.8
  - Using sixteen (16) threads:
    - Execution time: 2.2 seconds*
    - **Speedup** = 14 / 1.7 = 8.2

- Ten million (10.000.000) passwords: 17 seconds*
  - Using five threads:
    - Execution time: 39 seconds*
    - **Speedup** = 132 / 39 = 3.3
  - Using ten (10) threads:
    - Execution time: 22 seconds*
    - **Speedup** = 132 / 22 = 6
  - Using sixteen (16) threads:
    - Execution time: 17 seconds*
    - **Speedup** = 132 / 17 = 7.7

- One hundred million (100.000.000) passwords:
  - Using sixteen (16) threads:
    - Execution time: 141 seconds*
    - **Speedup** = 953 / 141 = 6.75

** These values are approximate*

6. Performance Analysis

While the parallel program generally exhibits significant speedup compared to the sequential program, it is noteworthy that for tests involving up to 100,000 passwords, the sequential program performs better in terms of execution time. However, as the number of passwords increases, the speedup slightly decreases. Nevertheless, parallelization consistently provides substantial performance improvement, especially when utilizing multiple threads effectively.

7. Conclusion

The comparison between the sequential and parallel programs reveals a nuanced picture. While the parallel program offers considerable speedup for larger datasets, it is outperformed by the sequential program for smaller datasets containing up to 100,000 passwords. Therefore, the choice between sequential and parallel implementations should be made judiciously, considering the dataset size and the specific requirements of the application.

8. Recommendations

Given the mixed performance observed across different dataset sizes, it is recommended to evaluate the characteristics of the dataset before deciding on the execution model. For smaller datasets (up to 100,000 passwords), the sequential program may offer better performance, whereas for larger datasets, transitioning to the parallel program can lead to significant speedup. Additionally, further optimizations and fine-tuning of parallelization parameters are advised to maximize performance across varying dataset sizes.