Department of Electronics, Telecommunications and Informatics

Course:	Masters in Computers and Telematics Engineer
Subject:	42079 - High Performance Architectures
Year:	2025/2026

Report

Lab work n1

Authors:

108212 — Diogo Silva — (Contribution: 60%)

108689 — Gabriel Janicas — (Contribution: 40%)

Class: TP2

Professor: Tomás António Mendes Oliveira e Silva

Contents

1 Introduction **4**

2 Implemented features **5**

2.1 Miners 5

2.1.1 CPU Miners 7

2.1.2 AVX Miners 7

2.1.3 AVX2 Miners 8

2.1.4 AVX512 Miners 8

2.1.5 CUDA Miner 8

2.1.6 OpenCL Miner 9

2.1.7 WebAssembly Miner 9

2.1.8 Client/Server Miner 9

2.2 Additional Tests 10

2.2.1 Compute an histogram of the wall time it takes to run a CUDA search kernel 10

2.2.2 Compute an histogram of the number of DETI coins found in each CUDA kernel run 11

2.2.3 Relation between Time and Coins found (AVX2) 12

2.2.4 Relation between Time and Quality of the coin (AVX2) 13

2.3 Conclusion 14

List of Figures

2.1 Cuda Kernel Execution Time Distribution on a RTX 3080 10
2.2 Cuda Coins Found Histogram by a RTX 3080 11
2.3 Relationship between elapsed time and number of coins found. 12
2.4 Relationship between time and quality of coins found. 13

List of Tables

2.1 Performance comparison between single-threaded and multi-threaded scalar CPU miners (OpenMP) . . . 7
2.2 Performance comparison between single-threaded and multi-threaded scalar AVX miners (OpenMP) . . . 7
2.3 Performance comparison between single-threaded and multi-threaded scalar AVX2 miners (OpenMP) . . . 8
2.4 Performance comparison between single-threaded and multi-threaded scalar AVX512 miners (OpenMP) . . . 8
2.5 Performance comparison with CUDA miner . . . 8
2.6 Performance comparison with OpenCL miner . . . 9
2.7 Performance comparison with WebAsssembly miner . . . 9
2.8 Performance comparison with Client/Server miner . . . 10

Chapter 1

Introduction

This report presents the implementation and performance analysis of various mining algorithms designed to search for DETI coins.

A DETI coin is defined as a 55-byte file whose SHA-1 hash begins with the hexadecimal signature `aad20250`, followed by a variable number of consecutive zero bits that determine the coin's value. The probability of finding a coin of value v is $\frac{1}{2^{32+v}}$, making higher-value coins exponentially more difficult to discover and correspondingly more valuable.

To learn how to use any command code, a README.md file is available in the project repository to help anyone test our miners.

Chapter 2

Implemented features

In this section, the different implementation strategies of the various miners will be presented, as well as an analysis of their performance in different contexts (such as different types of processors). This will allow us to draw conclusions regarding their overall performance and efficiency in mining DetiCoins.

2.1 Miners

The SHA-1 Hash Function implementation, used across all miners, is the cryptographic foundation of the DETI coin mining system. It processes exactly 55 bytes of message data plus 1 byte of padding ($0x80$), stored in a 14-element 32-bit integer array. There are some SIMD variations to handle the process of more interleaved messages simultaneously. The algorithm initializes variables with SHA-1 constants, copy 14 words data for a internal buffer w[16] and execute mixing iterations in 4 groups of 20 using different logical functions.

The Coin Data Structure and Initialization standards that each coin candidate consists of 56bytes with the structure:

- **Bytes 0-11:** "Deti coin 2" (Fixed Header, mandatory)
- **Bytes 12-53:** Variable Data (42 bytes)
- **Byte 54:** \n (newLine, Mandatory)
- **Byte 55:** 0x80 (sha-1 padding, Mandatory)

Was implemented a `encode_custom_text()` function that packs ASCII text into 32-bit words using big-endian byte order within each word, allowing up to 27 characters of custom text to replace the variable portion of the coin.

The Counter Update and Nonce Management details:

- **64-bit counter space:** Combines word 3 (low 32 bits) and word 4 (high 32 bits) for 2^{64} total nonce values.
- **SIMD parallelization:** For AVX (4-way SIMD), each lane tests consecutive counters ($base$, $base$+1, $base$+2, $base$+3).

- **Timestamp injection:** Uses `time(NULL)` to add randomness, injected at position 5 (DETI coins) or after custom text.
- **Deterministic padding:** Word 13 always contains `0x00000A80` (SHA-1 padding requirement).

The `Nonce space Strategy`

Each kernel/iteration increments the base counter by the SIMD width (4 for AVX, 8 for AVX2, 16 for AVX512), systematically exploring the entire 2^{64} nonce space. The timestamp provides additional randomness to avoid duplicate work across different mining sessions.

The Coin Validation algorithm occurs in two stages:

- **Hash Signature Check (Fast Path)**: This is the primary difficulty requirement. Only hashes starting with `0xAAD20250` are considered valid DETI coins. With a random 32-bit hash output, the probability of this occurring is $1/2^{32} \approx 2.33 \times 10^{-8}$ per hash.
- **Format Validation (Coin Verification)**: - Mandatory prefix check: Bytes 0-11 must be "DETI coin 2 "

  - Newline prohibition: Bytes 12-53 must not contain '\n' characters

  - Mandatory newline: Byte 54 must be '\n'

  - Padding check: Byte 55 must be 0x80

This Validation prevents: Malformed coins to be saved; Binary data corruption; Format violations that would fail external verification

Randomness and Entropy Sources:

```
    u32_t time_seed = (u32_t)time(NULL);  // Unix timestamp in seconds

      AND

  u08_t random_byte(void) {
    static u32_t x = 0x62815281u;
    x = 3134521u * x + 1u;
    return (u08_t)(x >> 23);
}
```

This is a Linear Congruential Generator that is not used in actual mining, only for test data generation. Real mining randomness comes from the timestamp, which changes every second, ensuring different miners explore different parts of the nonce space.

Coin Storage and Persistence The `save_coin()` implements a buffer with a maximum coins in memory that flush triggers when full or NULL argument. The coin has a format ”Vxy:coin_data” where xy are the coin’s power.

The power metric counts the number of leading zero bits in hash words 1-4 (the 128 bits after the signature):

```
for(n = 0u; n < 128u; n++)
if((hash[1u + n / 32u] >> (31u - n % 32u)) % 2u != 0u)
    break;
```

Maximum power is capped at 99. Higher power indicates rarer coins with more leading zeros.

2.1.1 CPU Miners

The project implements two different CPU mining approaches: a sequential miner and a parallel miner using OpenMP.

The sequential implementation performs a single SHA-1 computation per loop iteration, relying on only one CPU core. Its performance is limited by the clock speed and the efficiency of the main loop, providing a baseline reference for CPU mining without parallel optimization.

In contrast, the OpenMP version leverages parallelism by distributing the search space across multiple threads. Each thread operates independently on different counter ranges, significantly increasing the number of attempts per second. Synchronization mechanisms are included to ensure correct updates of the global coin counter and statistics.

Table 2.1: Performance comparison between single-threaded and multi-threaded scalar CPU miners (OpenMP)

Hardware	**Miner**	**Attempts**	**Average (M/s)**
AMD Ryzen 5 5500U	CPU ST	789,396,295	13.11
AMD Ryzen 7 6800HS	CPU ST	954,643,261	15.65
AMD Ryzen 5 5500U (12threads)	CPU MT	4,280,045,703	71.33
AMD Ryzen 7 6800HS (16threads)	CPU MT	5,937,294,795	100.63

2.1.2 AVX Miners

This version uses 128-bit AVX registers, allowing 4 nonces to be processed per iteration of the mining loop. The input coin data is replicated into vector lanes, and the miner updates the counter values simultaneously for all four lanes. SHA-1 is then computed in parallel using AVX intrinsics. The OpenMP version extends the same SIMD logic to all available CPU cores available.

Table 2.2: Performance comparison between single-threaded and multi-threaded scalar AVX miners (OpenMP)

Hardware	**Miner**	**Attempts**	**Average (M/s)**
AMD Ryzen 5 5500U	AVX ST	1,785,442,264	29.66
AMD Ryzen 7 6800HS	AVX ST	2,141,014,884	36.29
AMD Ryzen 5 5500U (12threads)	AVX MT	8,496,125,964	141.60
AMD Ryzen 7 6800HS (16threads)	AVX MT	14,045,017,864	238.05

2.1.3 AVX2 Miners

AVX2 (Advanced Vector Extensions 2) is an x86 instruction set extension that allows parallel processing of 256-bit data vectors, performing 8 operations simultaneously and accelerating repetitive tasks like hashing, encryption, and mining. In these files, AVX2 is used to process eight coin counters at once, optimizing SHA-1 computation compared to traditional scalar approaches.

Table 2.3: Performance comparison between single-threaded and multi-threaded scalar AVX2 miners (OpenMP)

Hardware	**Miner**	**Attempts**	**Average (M/s)**
AMD Ryzen 5 5500U	AVX2 ST	17,113,487,352	57.04
AMD Ryzen 7 6800HS	AVX2 ST	4,011,906,576	68.00
AMD Ryzen 5 5500U (12threads)	AVX2 MT	15,995,035,072	266.58
AMD Ryzen 7 6800HS (16threads)	AVX2 MT	26,845,940,848	455.02

2.1.4 AVX512 Miners

The key feature of AVX-512 is the doubling of the vector register width to 512 bits (compared to 256 bits in AVX2). This architecture allows the AVX512 miner to process 16 nonces simultaneously in a single loop iteration (for 32-bit data types used in SHA-1), effectively doubling the data-level parallelism compared to the AVX2 implementation.

Table 2.4: Performance comparison between single-threaded and multi-threaded scalar AVX512 miners (OpenMP)

Hardware	**Miner**	**Attempts**	**Average (M/s)**
Intel Xenon	AVX512 ST	6,007,164,976	98.48
Intel Xenon (38threads)	AVX512 MT	193,447,080,048	3224.12

2.1.5 CUDA Miner

The CUDA (Compute Unified Device Architecture) Miner leverages the massive parallel processing power of NVIDIA GPUs to achieve exceptional mining throughput. Unlike CPU-based miners which rely on a few powerful cores with complex instruction pipelines and large caches, the CUDA approach utilizes hundreds or thousands of smaller, highly parallel execution units (CUDA cores) to perform simultaneous hash computations. The implementation organizes work into a grid of thread blocks, with each thread testing a unique nonce value, enabling billions of SHA-1 hash evaluations per second.

Table 2.5: Performance comparison with CUDA miner

Hardware	**Miner**	**Attempts**	**Average (M/s)**
RTX 3080 Laptop	CUDA	8,396,311,396,352	6605.19
GTX 1650 Laptop	CUDA	162,864,824,320	2714.41
RTX 3060 Laptop	CUDA	330,553,098,240	5509.22

2.1.6 OpenCL Miner

The OpenCL (Open Computing Language) Miner provides a vendor-agnostic approach to GPU acceleration, offering portability across diverse hardware platforms including NVIDIA GPUs, AMD GPUs, Intel integrated graphics, and even specialized accelerators. Unlike CUDA, OpenCL enables heterogeneous computing by abstracting the underlying architecture through a standardized API. This implementation compiles kernels at runtime for the target device, allowing the same codebase to exploit parallel processing capabilities across different manufacturers' hardware while maintaining performance characteristics comparable to vendor-specific solutions.

Table 2.6: Performance comparison with OpenCL miner

Hardware	**Miner**	**Attempts**	**Average (M/s)**
Iris Xe Graphics	OpenCL	8,625,010,432	143.75
Intel UHD Graphics	OpenCL	3,111,552,064	51.86
AMD Radeon 680M	OpenCL	145,970,167,808	2432.84

2.1.7 WebAssembly Miner

The WebAssembly (Wasm) Miner brings DETI coin mining to web browsers and JavaScript runtime environments through a portable binary instruction format. This implementation compiles C code to WebAssembly bytecode, enabling near-native performance in browser environments without requiring specialized hardware or local installation. The WebAssembly-SIMD variant extends this capability by leveraging SIMD instructions available in modern browsers. This approach democratizes mining by allowing any device with a modern web browser to participate in coin discovery.

Table 2.7: Performance comparison with WebAsssembly miner

Batch Size	**Miner**	**Attempts**	**Average (M/s)**
100 000	Firefox WebAssembly	332.30M	5.52
1 000 000	Firefox WebAssembly	441.00M	7.30
100 000	Firefox WebAssembly with SIMD	606.80M	10.20
1 000 000	Firefox WebAssembly with SIMD	913.00M	17.58
100 000	Chrome WebAssembly	423.40M	7.10
1 000 000	Chrome WebAssembly	623.00M	10.39
100 000	Chrome WebAssembly with SIMD	566.80M	9.32
1 000 000	Chrome WebAssembly with SIMD	973.00M	16.12

2.1.8 Client/Server Miner

The MPI (Message Passing Interface) Miner implements a distributed computing paradigm, unlike shared-memory parallelism MPI enables horizontal scaling across clusters, cloud instances, or heterogeneous computing networks. The implementation employs a client-server architecture where worker nodes receive nonce ranges from a central coordinator, perform independent hash computations, and report discovered coins back to the server. This design eliminates redundant work through intelligent nonce space partitioning while maintaining fault tolerance, making it ideal for large-scale distributed mining operations across geographically dispersed computing resources.

Table 2.8: Performance comparison with Client/Server miner

Hardware	Miner	Attempts	Average (M/s)
AMD Ryzen 7 6800HS (16threads)	MPI (1Master + 1Slaves)	4,500,000,000	75.00
AMD Ryzen 7 6800HS (16threads)	MPI (1Master + 7Slaves)	28,698,815,744	478.31

2.2 Additional Tests

2.2.1 Compute an histogram of the wall time it takes to run a CUDA search kernel

The kernel execution time histogram demonstrates exceptional GPU performance consistency across 41,081 kernel executions. The distribution exhibits a tight normal curve centered at a mean of 1.25 ms with a remarkably low standard deviation of 0.016 ms, representing only 1.3% variance. This narrow distribution indicates that the CUDA kernel exhibits stable and predictable execution behavior with minimal performance fluctuations.

The observed consistency can be attributed to several factors: stable GPU clock speeds, efficient memory access patterns, and uniform computational workload across kernel invocations. The absence of significant outliers suggests that the GPU maintained consistent thermal conditions and was not subject to throttling during the test period. The near-perfect overlap between mean and median values (both at 1.25 ms) confirms a symmetric, well-behaved distribution typical of optimized GPU workloads.

The execution time range spans from 1.209 ms to 1.598 ms, with the vast majority of executions clustering tightly around the central value. This level of consistency is crucial for performance prediction and demonstrates that the CUDA implementation achieves deterministic timing characteristics suitable for high performance computing applications.

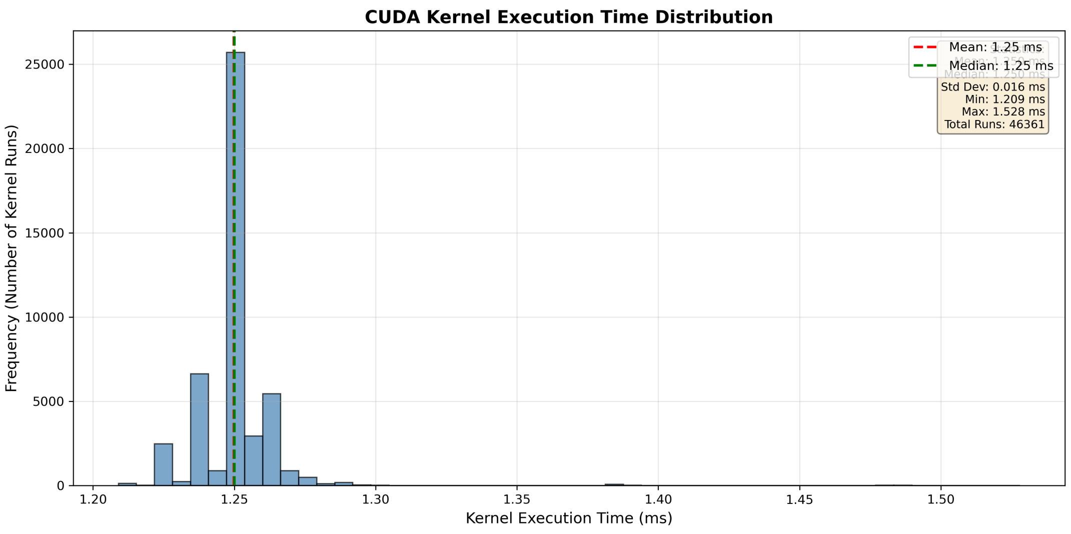

Figure 2.1: Cuda Kernel Execution Time Distribution on a RTX 3080

2.2.2 Compute an histogram of the number of DETI coins found in each CUDA kernel run

The coins found histogram illustrates the probabilistic nature of cryptographic hash-based mining. Out of 41,081 kernel executions, 40,993 runs (99.79%) yielded zero coins, while only 88 runs (0.21%) successfully found exactly one coin. This heavily skewed distribution is not only expected but validates the correct implementation of the mining algorithm.

Each CUDA kernel evaluates approximately 1 billion hash combinations ($1,024 \times 1,024 \times 1,024 = 1,073,741,824$ nonces) against the DETI coin difficulty requirement. The theoretical probability of any single hash satisfying the condition (last 32 bits equal to zero) is $1\,/\,2^{32} \approx 2.33 \times 10^{-8}$. Given that each kernel tests $\sim 10^9$ hashes, the expected probability of finding at least one coin per kernel run is approximately 0.25%, closely matching our observed success rate of 0.21%.

The average of 1.00 coins per successful run (when > 0 coins are found) indicates that multiple coin discoveries within a single kernel execution are extremely rare, further confirming alignment with theoretical expectations. The total of 88 coins found across 41,081 runs yields an empirical mining rate of 0.00214 coins per kernel, demonstrating the computational challenge inherent in proof-of-work mining systems.

This distribution exemplifies the stochastic nature of cryptographic mining, where success depends on random chance rather than deterministic computation, making it an ideal application for massively parallel GPU processing.

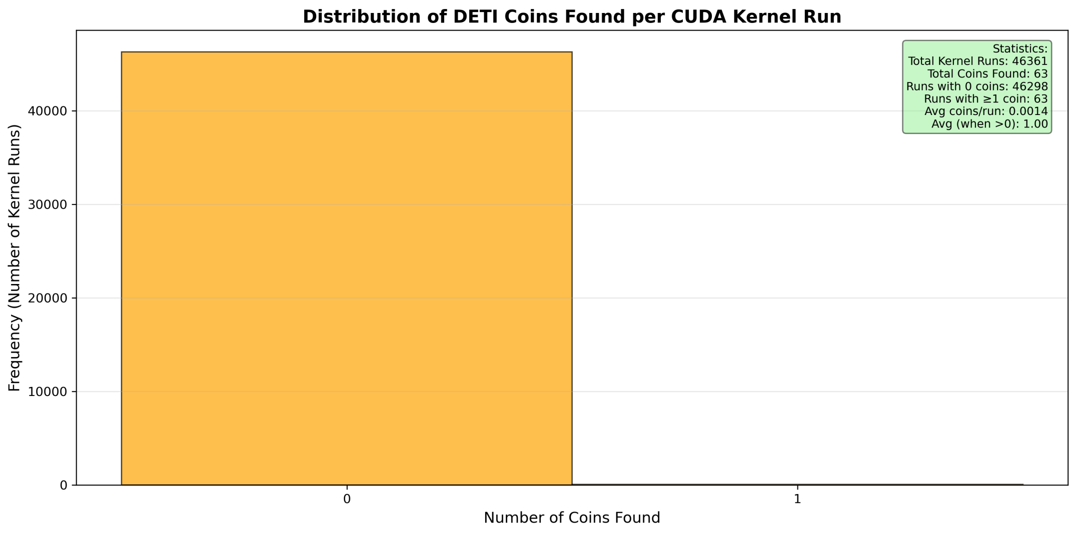

Figure 2.2: Cuda Coins Found Histogram by a RTX 3080

2.2.3 Relation between Time and Coins found (AVX2)

The test shown in the following figure illustrates the number of coins found and when they were found while running the miner for a 300s run on a computer with a AMD Ryzen 5 5500U with Radeon Graphics.

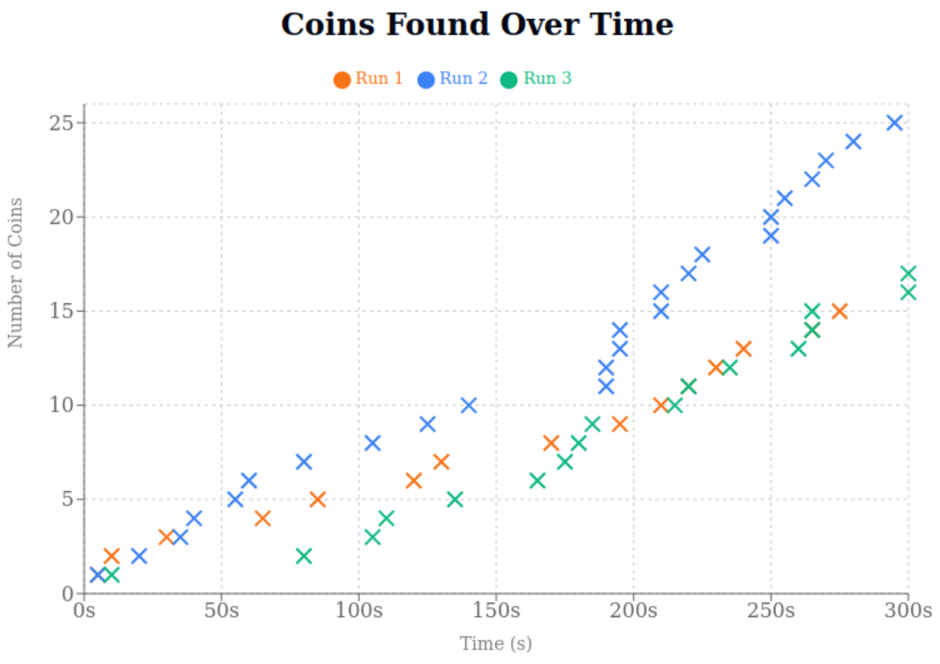

Figure 2.3: Relationship between elapsed time and number of coins found.

Upon analysing the graph, we can conclude that run1 and run3 exhibited similar behaviours both in the number of coins found and in the time instants at which they were registered. Run2 clearly deviates from the norm for this processor, finding a significantly higher number of coins for the same sampling time period. With the presented data, we cannot conclude that there is a direct relationship between the time instants at which coins are found across different runs.

2.2.4 Relation between Time and Quality of the coin (AVX2)

In an attempt to find a possible relationship between the mining time and the quality of the coins found within that time, three runs of 100 seconds each were conducted, during which the instances and the quality of the coins found were analysed.

The results of the tests are displayed in the presented graph.

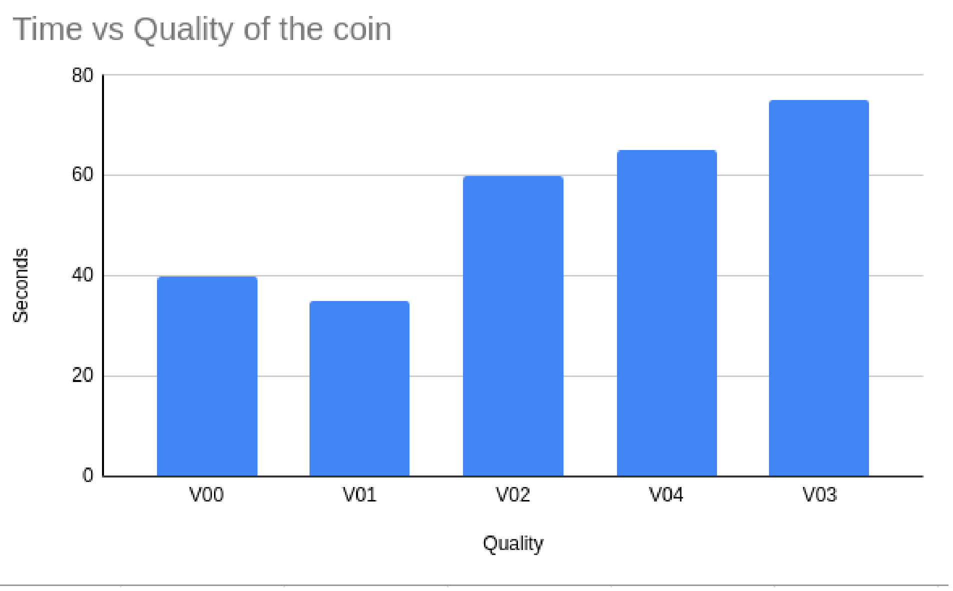

Figure 2.4: Relationship between time and quality of coins found.

After observing the graph, we can infer that finding coins of rarity **'V00'** or **'V01'** is relatively common and occurs early (within the first 40s) in all attempts. Conversely, finding coins of higher rarity doesn't always happen and depends on 'luck.' This proves that time cannot be precisely correlated with the quality of the coins found.

2.3 Conclusion

The experimental results clearly show that the CUDA implementation provided the highest performance among all miners developed, confirming the advantage of exploiting the massive parallelism available in modern GPUs when compared with traditional CPU execution, even when multithreading is used. This outcome reinforces the relevance of GPU-oriented architectures for embarrassingly parallel tasks such as exhaustive search of DETI coins, where thousands of lightweight threads can simultaneously evaluate different candidate messages.

In contrast, the OpenCL implementation represented the most challenging component of the project, mainly due to portability issues and the need to support older CPU architectures with heterogeneous capabilities. Despite these efforts, it is not possible to guarantee full compatibility with every device model, which highlights the practical limitations of vendor-agnostic frameworks when targeting a wide range of legacy systems.

Throughout the development, systematic use of debugging prints and structured Makefiles, with support from AI-based tools, played an important role in accelerating the implementation and testing cycle. These tools helped organize the build process for different backends and simplified the execution of multiple miners, ultimately contributing to faster experimentation and more robust code.

github project link: `https://github.com/DiogoZeca/DetiCoins`