SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza Università di Roma

Computer Science

Decolorizing images using AWS cloud services

Students:
Davide Paossi 1950062
Giuseppe Carnicella 1950329
Simone Giovagnoni 1932180

Professor:
Emiliano Casalicchio

Cloud Computing Project Report

Cloud Computing - A.Y. 2024/2025

1 Introduction

This project presents a serverless image processing system developed using AWS Lambda. The system takes an image and a specific color as input, and returns a transformed version of the image where all colors are converted to grayscale except for the selected one, which is preserved in its original hue. This technique, often referred to as *selective colorization*, is commonly used in photography and design to draw attention to a specific element or highlight important details.

The solution leverages AWS Lambda to ensure scalability, cost efficiency, and ease of deployment. By combining cloud computing with image manipulation techniques, the project demonstrates how serverless architectures can be used to build lightweight, on-demand image processing applications.

The goal of this project is not only to implement the selective colorization functionality, but also to explore the capabilities of cloud-based image processing and assess the performance of AWS Lambda in this context.

2 Cloud Infrastructures

2.1 AWS Lambda

AWS Lambda is a serverless computing service that executes code in response to events and automatically manages the required computational resources. Developers can upload their code to Lambda, and the service only executes code when needed, reducing operating costs and simplifying application management.

The Lambda function executes the core image processing logic. It receives the image and the selected color as input, performs the grayscale transformation and color filtering, and returns the final result.

2.2 Amazon S3

Amazon S3 (Simple Storage Service) is a highly scalable, durable, and secure object storage service provided by Amazon Web Services (AWS). It is designed to store and retrieve any amount of data from anywhere on the web, making it a fundamental building block for many cloud-based applications and services.
Acts as a storage layer for both the input and output images. Users upload the original image to an S3 bucket, and the processed image is saved to the same or another bucket after the transformation.

2.3 CloudWatch

The system also uses Amazon CloudWatch to monitor performance, resource usage, and operational health on the server side. CloudWatch is a native AWS monitoring service that collects logs, metrics, and events from AWS resources, making it an essential tool for analyzing the behavior and efficiency of serverless applications.
In our case, the cloudwatch metrics used are:

- Invocation
- Error Count and Success rate
- Total Concurrent Executions
- Duration
- Throttles

3 Application

3.1 Image Input and Color Selection

The process begins when a user uploads an image (typically in JPEG or PNG format) and specifies a target color they wish to retain. This color can be provided in standard RGB format.
The input image is stored in an Amazon S3 bucket, and the Lambda function is triggered either automatically (via an S3 event) or through a direct API call.

3.2 Grayscale conversion

The first transformation step is to create a grayscale version of the image applying the luminance formula to each pixel.

$$Gray = 0.299 * R + 0.587 * G + 0.114 * B \tag{1}$$

This formula ensures that the grayscale image maintains a perceptually balanced brightness. The red, green, and blue values of each pixel are replaced by the same grayscale intensity, effectively removing all the color from the image.

3.3 Color selection

Then, on the original colored image, the system checks the distance of the color of each pixel from the selected color, if it is under the threshold, the pixel's color is maintained. This technique allows the system to preserve not only exact matches of the selected color but also close shades, depending on the threshold level set in the algorithm.

3.4 Output image

After processing all pixels, the final image is now mostly black and white with a splash of the chosen color. This image is then saved as a new file and uploaded to the Amazon S3 bucket, from where the user can download or visualize it.

4 Results

4.1 Locust

Locust provided a powerful and flexible way to simulate real-world usage scenarios and test the robustness of our serverless system. It helped ensure that the image processing pipeline is not only functional but also scalable and resilient under stress—critical qualities for any production-grade cloud application.

- **Measure Lambda Execution Time**

  We assessed how long the Lambda function takes to process images of varying sizes and resolutions under different levels of concurrent load.

- **Evaluate System Scalability**

  By gradually increasing the number of simulated users, we observed how well the system scales and whether any bottlenecks occur.

- **Identify Failure Points**

  We monitored the behavior of the system when stressed beyond typical usage (e.g., rapid bursts of requests) to identify potential failure scenarios.

4.2 Test Results

In this section, we evaluate the outcomes presented in Section 4.3.
We began by assigning the lowest possible memory configuration available for AWS Lambda functions.
While the results for small images were promising (see Section 4.3.1), the function was unable to process 4K images due to memory limitations.
We investigated this issue by incrementally increasing the memory allocation to 128MB (Section4.3.2), 300MB (Section4.3.3), and 512MB (Section4.3.4). Despite these increases, the function still could not consistently handle 4K images.
At 1400MB, the function became viable for most 4K images, achieving approximately 92% availability (Section4.3.6).
However, timeout issues persisted when the memory was raised to 1700MB (Section4.3.8).
Only when allocating 2000MB of memory did we reach our target of 100% availability for 4K image processing (Section4.3.10).

At this configuration (2000MB), the `decolor` function processes:

- Small images in an average of 113ms, using approximately 160MB of memory
- 4K images in about 3 seconds, consuming around 1800MB of memory

In terms of throughput per individual concurrent execution:

- Small images are processed at an average rate of 530 images per minute
- 4K images at approximately 19.5 images per minute

For 4K images, the primary performance bottlenecks are the `apply_mask` and `compute_color_distance` functions. Additionally, the conversion from PIL to NumPy format contributes significantly to the processing time.

In the case of small images, the main bottleneck is the retrieval of the image from the S3 bucket, which accounts for roughly half of the total execution time.

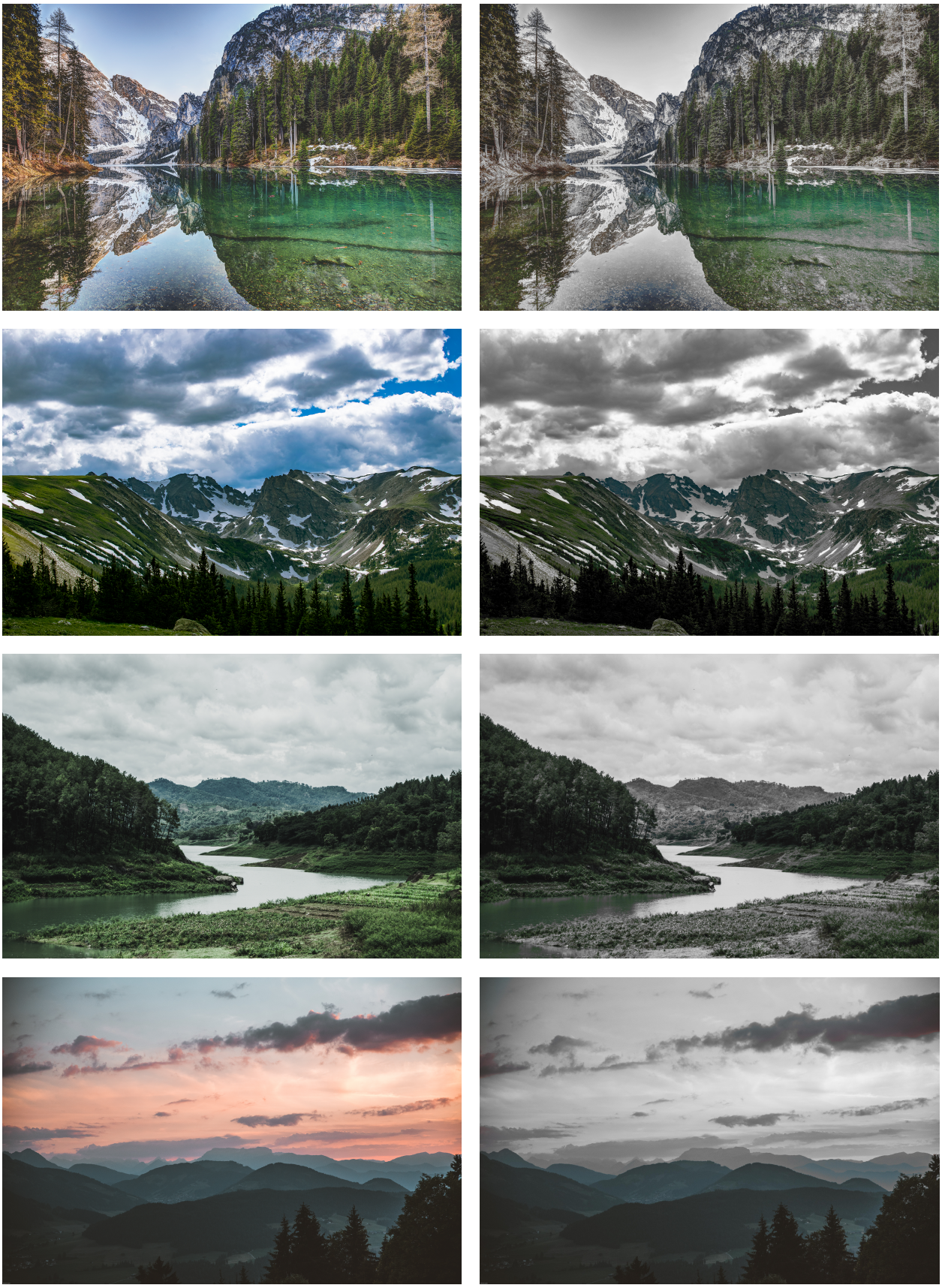

Figure 1: Original and decolored versions of the images with different thresholds

4.3 Experimental Results

4.3.1 Small images with 128MB

Time usage for each function
apply_mask : 299.733*ms*
compute_color_distance : 333.762*ms*
decolor : 865.450*ms*
get_image : 84.861*ms*
save_to_buffer : 30.901*ms*
transform_image_from_PIL_to_np : 84.276*ms*
transform_image_to_PIL : 17.184*ms*
upload_to_s3 : 92.681*ms*

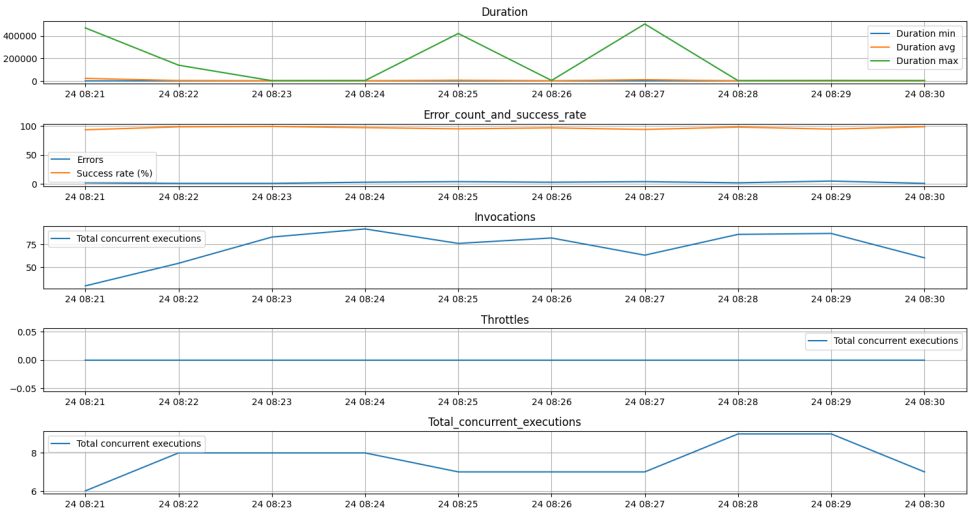

Figure 2: CloudWatch Metrics for small images with 128MB

4.3.2 4k images with 128mb

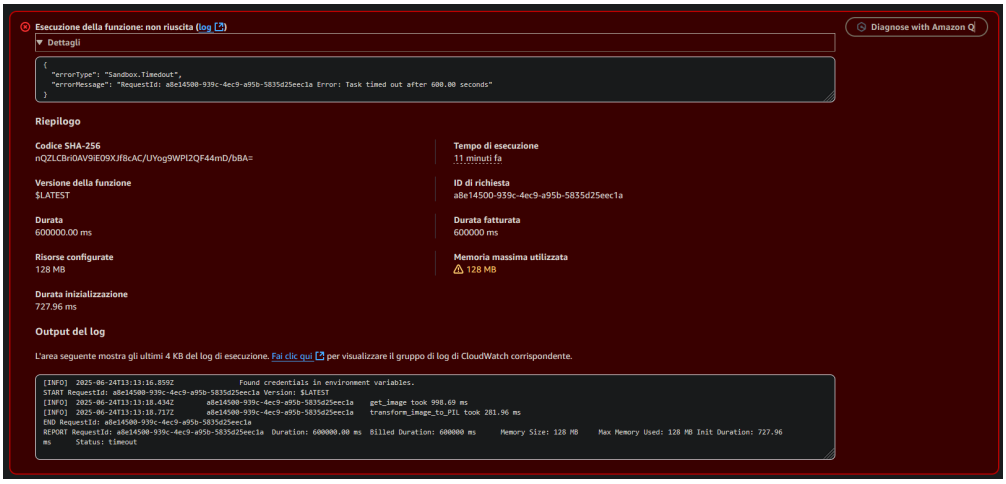

Figure 3: Lambda timeout for 4k images with 128MB

4.3.3 4k images with 300mb

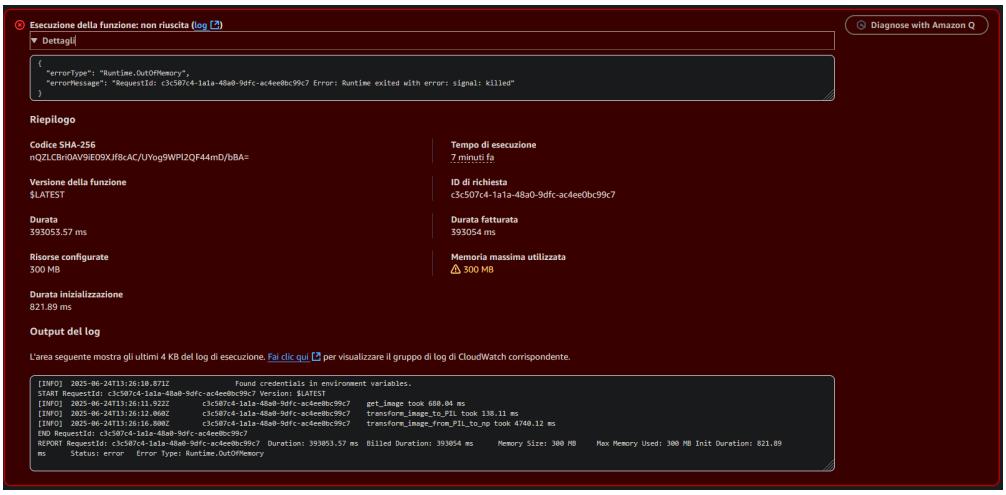

Figure 4: Lambda timeout for 4k images with 300MB

4.3.4 4k images with 512mb

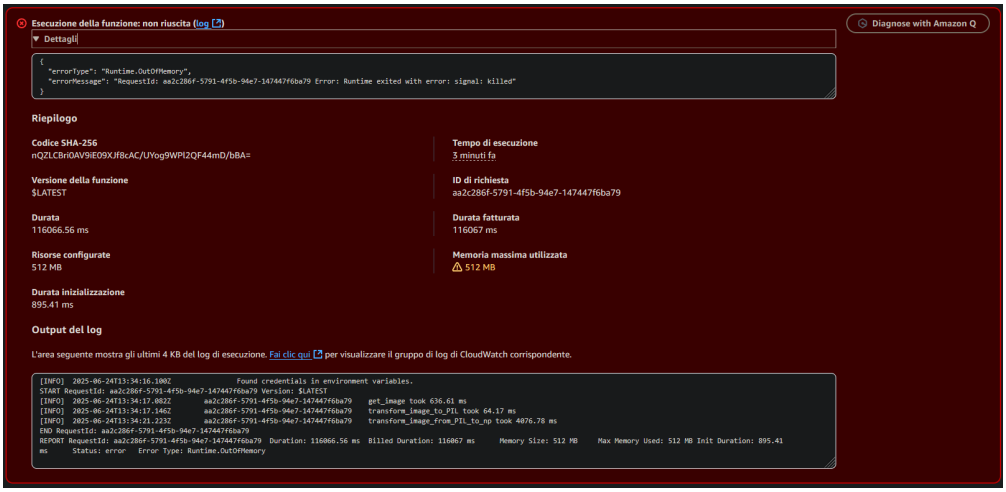

Figure 5: Lambda timeout for 4k images with 512MB

4.3.5 Small images with 1400MB

Time usage for each function
$apply\_mask$: 509.311ms
$compute\_color\_distance$: 463.589ms
$decolor$: 1418.169ms
get_image : 161.327ms
$save\_to\_buffer$: 38.570ms
$transform\_image\_from\_PIL\_to\_np$: 328.594ms
$transform\_image\_to\_PIL$: 0.760ms
$upload\_to\_s3$: 82.758ms

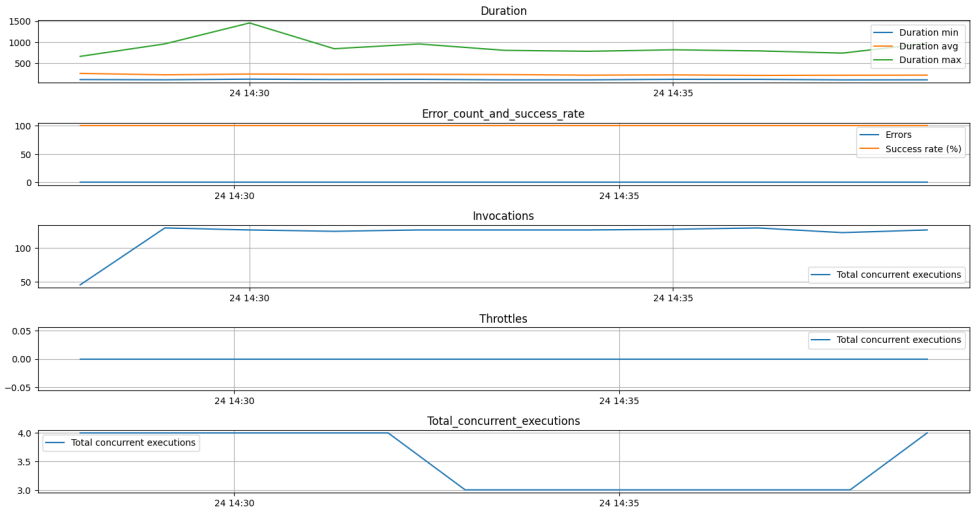

Figure 6: CloudWatch metrics for small images with 1400MB

4.3.6 4K images with 1400MB

Time usage for each function
$apply\_mask$: 1420.095ms
$compute\_color\_distance$: 1291.503ms
$decolor$: 3858.767ms
get_image : 370.891ms
$save\_to\_buffer$: 106.215ms
$transform\_image\_from\_PIL\_to\_np$: 902.313ms
$transform\_image\_to\_PIL$: 1.723ms
$upload\_to\_s3$: 120.542ms

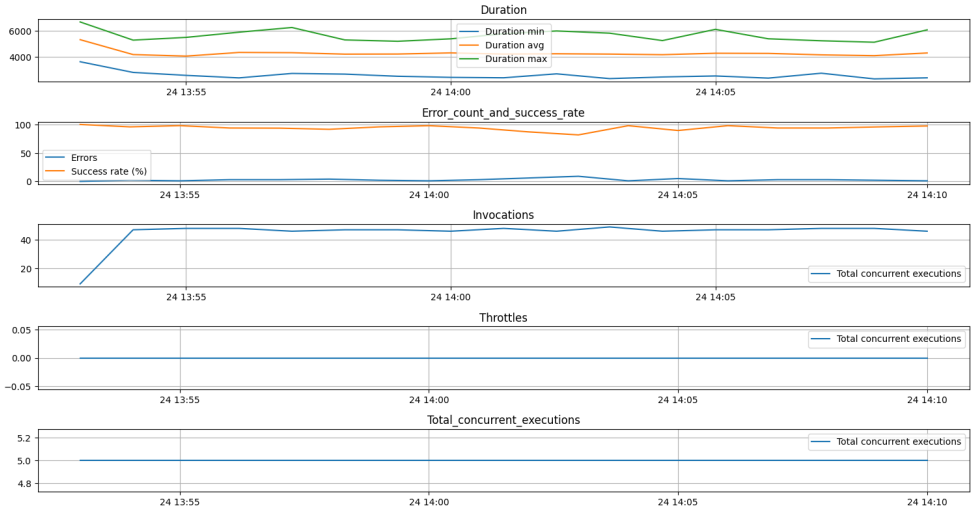

Figure 7: CloudWatch Metrics for 4K images with 1400MB

4.3.7 Small images with 1700MB

Time usage for each function
$apply\_mask$: 26.556ms

$compute\_color\_distance$: 26.184ms
$decolor$: 112.890ms
get_image : 37.859ms
$save\_to\_buffer$: 2.585ms
$transform\_image\_from\_PIL\_to\_np$: 6.662ms
$transform\_image\_to\_PIL$: 0.216ms
$upload\_to\_s3$: 47.768ms

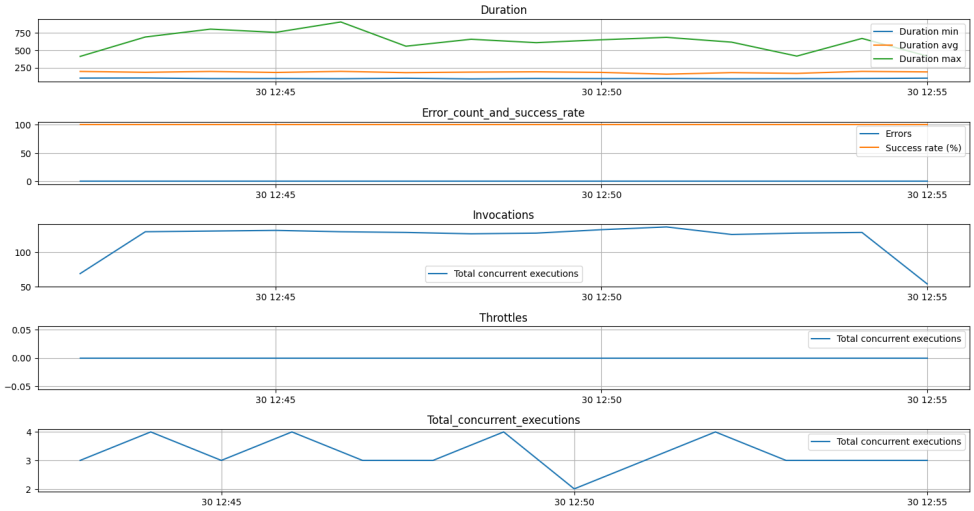

Figure 8: CloudWatch Metrics for Small images with 1700MB

4.3.8 4K images with 1700MB

Time usage for each function
$apply\_mask$: 1224.853ms
$compute\_color\_distance$: 1115.620ms
$decolor$: 3324.605ms
get_image : 377.605ms
$save\_to\_buffer$: 90.128ms
$transform\_image\_from\_PIL\_to\_np$: 759.747ms
$transform\_image\_to\_PIL$: 0.654ms
$upload\_to\_s3$: 113.012ms

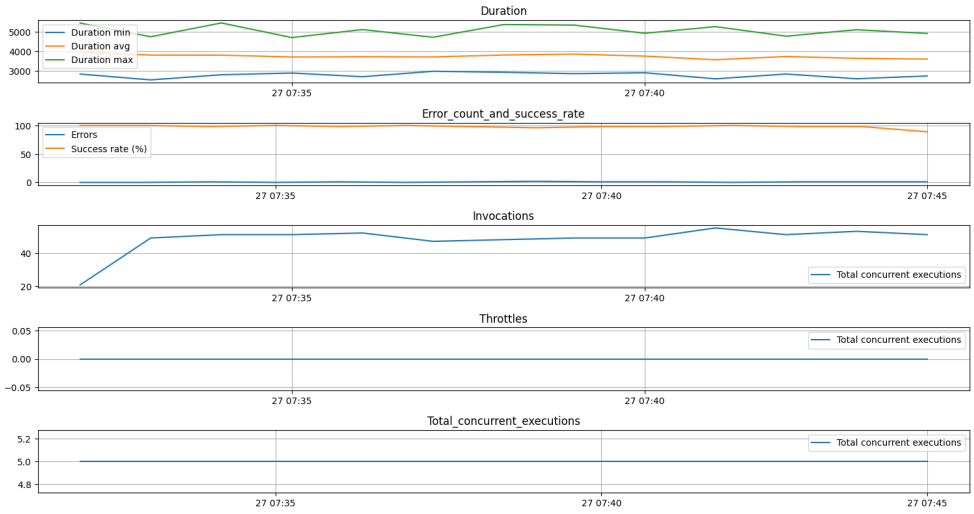

Figure 9: CloudWatch Metrics for 4K images with 1700MB

4.3.9 Small images with 2000MB

Time usage for each function
$apply\_mask$: 23.390ms
$compute\_color\_distance$: 22.879ms
$decolor$: 113.149ms
get_image : 43.008ms
$save\_to\_buffer$: 2.304ms
$transform\_image\_from\_PIL\_to\_np$: 6.326ms
$transform\_image\_to\_PIL$: 0.191ms
$upload\_to\_s3$: 55.656ms

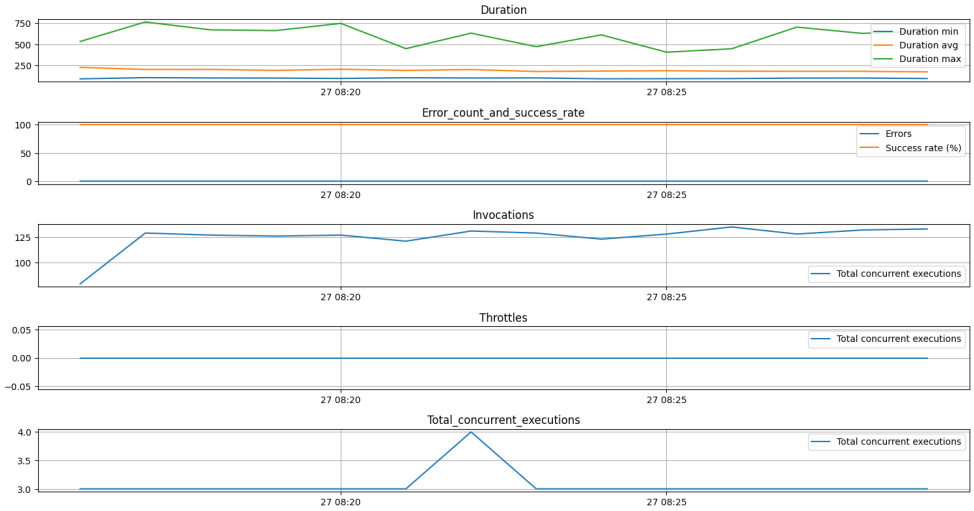

Figure 10: CloudWatch Metrics for small images with 2000MB

4.3.10 4K images with 2000MB

Time usage for each function
$apply\_mask$: 1128.576ms

$compute\_color\_distance$: 1021.906ms
$decolor$: 3070.871ms
get_image : 333.697ms
$save\_to\_buffer$: 83.184ms
$transform\_image\_from\_PIL\_to\_np$: 695.496ms
$transform\_image\_to\_PIL$: 0.316ms
$upload\_to\_s3$: 112.521ms

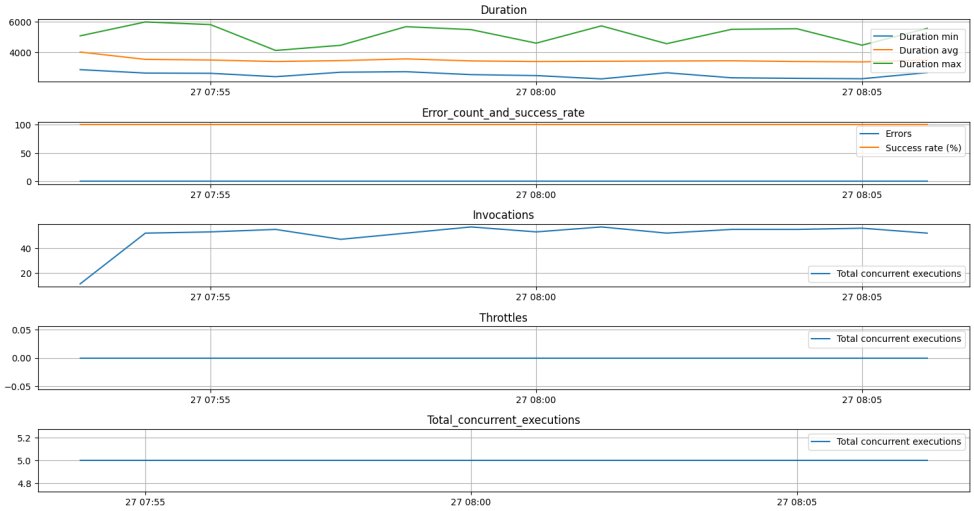

Figure 11: CloudWatch Metrics for 4K images with 2000MB

5 Conclusion

This project successfully demonstrates how serverless architectures can be leveraged to build efficient, scalable, and cost-effective image processing systems in the cloud. By utilizing a combination of AWS Lambda, Amazon S3, and Amazon CloudWatch, we designed and implemented a selective colorization pipeline capable of handling a variety of image sizes and operating under different memory configurations.

The system architecture ensures automatic scalability and responsiveness, allowing it to process image transformation requests on demand without the need for persistent servers or complex deployment pipelines. AWS Lambda handles the core image manipulation logic, triggered by user input or file uploads to S3, while CloudWatch provides logging and performance monitoring to support debugging and optimization.

Performance evaluations indicate that the solution remains responsive and efficient even when processing high-resolution images, with minimal latency and predictable resource usage. Moreover, the serverless nature of the architecture ensures that operational costs are incurred only when the system is in use, aligning well with modern principles of elastic and usage-based cloud computing.

Overall, the project validates the feasibility of using serverless technologies to address image processing challenges in a cloud-native manner, offering a flexible and maintainable approach that is well-suited to both academic exploration and real-world applications.