TCSS 562 Term Project Proposal

A Case Study on the Performance, Cost, and Architectural Implications of LLM-Generated Serverless Image Processing Pipelines

Bohan Xiong (bohanx2@uw.edu)
Xu Zhu (test@uw.edu)
Xiaoling Wei (test@uw.edu)

Instructor: Wes J. Lloyd

1 Cloud Application Description

Our team will implement an enhanced version of the predefined Image Processing Pipeline. This application will be built as an event-driven system on the AWS Lambda FaaS platform, which is known for its lightweight virtualization technology like Firecracker [1].

To conduct a deeper analysis of LLM code generation capabilities, our pipeline will consist of a sequential chain of five serverless functions. We have intentionally designed this pipeline to include functions with different computational loads (CPU-intensive vs. I/O-intensive) to evaluate LLM performance across varied workloads.

The pipeline will trigger on an S3 upload and execute the following logical steps:

1. **Function 1 (Greyscale):** Converts the input image to greyscale.
2. **Function 2 (Resize):** Resizes the image to a fixed resolution (e.g., 800x600).
3. **Function 3 (Color Depth Map):** Performs a CPU-intensive operation, such as mapping a 10-bit color depth image to 8-bit.
4. **Function 4 (Rotate):** Rotates the resized image by 90 degrees.
5. **Function 5 (Format Convert):** Performs an I/O-intensive operation, writing the final image as a different format (e.g., from JPEG to PNG) in the destination S3 bucket.

2 Design-Tradeoffs to be Investigated

Our case study will be two-dimensional, focusing on one primary and one secondary design trade-off.

2.1 Primary Investigation: LLM Comparison Theme

Our main goal is to compare the quality and efficiency of code generated by state-of-the-art Large Language Models. We will provide identical, detailed prompts for each of the five functions to three distinct LLMs:

- GPT-5 Thinking[1]
- Gemini 2.5 Pro
- DeepSeek V2.5 Research

We will instruct all LLMs to generate the function code in **Python 3.13**[2]. We chose this version as it is modern, well-supported by current AWS Lambda runtimes, and provides a relevant baseline for performance evaluation.

This will result in three functionally identical but implementation-distinct application versions. By including both CPU and I/O-intensive tasks, we can deeply analyze the differences in code optimization (e.g., algorithmic efficiency, I/O handling) among the LLMs.

2.2 Secondary Investigation: CPU Architecture

As an extension, we will investigate the impact of the underlying CPU architecture. We will deploy each LLM-generated pipeline version onto both available AWS Lambda architectures:

- Intel (x86_64)
- ARM (Graviton2)

This allows us to explore potential performance and cost differences (e.g., ARM's ~20% price discount) and observe if certain LLMs produce code that is better optimized for a specific architecture. Our investigation will build upon existing research in this area, such as the work by Chen et al. [2], which also compared these two architectures on AWS Lambda.

3 Proposed Evaluation Metrics

Our evaluation will compare all pipeline versions across performance and cost dimensions. We will use an identical set of input images and run repeated tests (e.g., 100+ runs) to gather statistical data.

- **Per-Function Runtime Analysis (Performance):** The average execution time (in milliseconds) for each individual function. This metric will be key to comparing how each LLM's code performs on CPU-intensive (Color Depth Map) vs. I/O-intensive (Format Convert) tasks.

[1] Should the LLM providers upgrade or change their models during the experiment (e.g., release a new version), we will use the corresponding upgraded model available at that time to ensure the relevancy of our study.

[2] If a newer Python environment is provided by the serverless platform at the time of the experiment, we will switch to the latest, such as the newly released Python 3.14.0.

- **Average Pipeline Turnaround Time (Performance):** The average end-to-end turnaround time (in seconds) for the complete five-function pipeline.
- **Hosting Cost (Cost):** The estimated total hosting cost (in $) for processing a large, hypothetical workload of 100,000 images for each LLM-generated pipeline version.
- **Cross-Architecture Performance (Perf/Cost):** A direct comparison of the runtime and cost for the same code (e.g., Gemini's) running on x86 vs. ARM.

For all performance metrics, we will report the statistical average, standard deviation, and coefficient of variation (CV).

4 Work Plan

Our team will use Git/GitHub for version control and collaborate remotely via Discord.

- **Phase 1 (Oct 22 - Oct 31): Feasibility Spike:** Manually create and test an AWS Lambda Layer containing the `Pillow` library. Deploy one test function (e.g., Greyscale) to confirm the dependency is correctly loaded. Following this validation, finalize prompts and generate all function code (5 functions x 3 LLMs) from all LLMs.
- **Phase 2 (Nov 1 - Nov 8):** Deploy all 6 pipeline versions (3 LLMs x 2 Architectures) on AWS Lambda.
- **Phase 3 (Nov 9 - Nov 16):** Develop test harness, potential adapting or using concepts from the SAAF framework mentioned in the course materials and drawing inspiration from established serverless benchmarks like SeBS [3], to execute experiments and collect all performance/cost data.
- **Phase 4 (Nov 17 - Nov 30):** Data analysis, graph generation, and creation of the final project report.

References

[1] Alexandru Agache, Marc Brooker, Andreea Florescu, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. 2020. Firecracker: Lightweight Virtualization for Serverless Applications. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*. USENIX Association, 419–434. https://www.usenix.org/conference/nsdi20/presentation/agache

[2] Xinghan Chen, Ling-Hong Hung, Robert Cordingly, and Wes Lloyd. 2023. X86 vs. ARM64: An Investigation of Factors Influencing Serverless Performance. In *24th ACM/IFIP International Middleware Conference (Middleware 2023), December 11-15, 2023, Bologna, Italy.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3631295.3631394

[3] Marcin Copik, Grzegorz Kwaśniewski, Maciej Besta, Michał Podstawski, and Torsten Hoefler. 2021. SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing. In *22nd In-*

ternational Middleware Conference (Middleware '21'). ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3464298.3476133