AWS Lambda TLQ Data Pipeline
Architectural Choices and Performance Evaluation

Halim Lee
School of Engineering and Technology
University of Washington
Tacoma, WA, USA
leeh10@uw.edu

Mathew Kim
School of Engineering and Technology
University of Washington
Tacoma, WA, USA
mattunho@uw.edu

Gregory Young Yi
School of Engineering and Technology
University of Washington
Tacoma, WA, USA

ABSTRACT

This paper details our team's implementation of an AWS Lambda TLQ (Transform, Load, Query) Data Pipeline, offering insights into key features, design trade-offs, and preliminary performance assessments. We compare the Switchboard Architecture against the Non-Switchboard Architecture, evaluating scalability, throughput, deployment cost/runtime, and cold/hot scaling performance. Leveraging the Sales Database, our pipeline includes three pivotal services: data extraction and transformation (Service #1), loading into a relational database (Service #2), and meaningful queries (Service #3). The implementation may showcase unique aspects, contributing to a nuanced exploration of TLQ application development. Our study sheds light on the implications of design choices, emphasizing the potential benefits of the Switchboard Architecture and addressing trade-offs associated with different database options.

KEYWORDS

AWS Lambda, TLQ Data Pipeline, Switchboard Architecture, Non-Switchboard Architecture, Scalability, Throughput, Deployment Cost, Cold and Hot Scaling Performance.

ACM Reference format:

Halim Lee, Matthew Kim, Gregory Young Yi. 2023. AWS Lambda TLQ Data Pipeline: Architectural Choices and Performance Evaluation. In *Proceedings of ACM Woodstock conference (WOODSTOCK'18). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/1234567890

1 Introduction

In response to escalating demands for efficient data processing, this report investigates the implementation and optimization of a serverless AWS Lambda TLQ Data Pipeline. The TLQ application facilitates the transformation, loading, and querying of sales data in CSV format. Our study delves into key features, design trade-offs, and initial performance assessments, with a focus on comparing the Switchboard Architecture and the Non-Switchboard Architecture. The comparative analysis evaluates aspects such as scalability, throughput, deployment cost/runtime, and cold/hot scaling performance.

Our interpretation of the TLQ application involves three pivotal services. Service #1 manages data extraction and transformation, introducing the Switchboard Architecture with various data transfer mechanisms. Service #2 efficiently loads transformed data into a relational database, considering options like Amazon Aurora or SQLite for performance and hosting costs. Service #3 executes meaningful queries on the loaded data, supporting aggregation and filtering. Potential unique aspects of our implementation, including specific programming languages, tools, or alternative cloud services, contribute to a nuanced exploration of TLQ application development.

To ensure clarity for readers unfamiliar with the TLQ application, insights are provided into the Sales Database, our chosen dataset. Acronyms are defined, and relevant references are incorporated. This report aims to offer valuable insights into serverless TLQ Data Pipelines, providing considerations for architectural choices, scalability, and performance optimization.

1.1 Research Questions

Q1. How do Switchboard and Non-Switchboard Architectures in AWS Lambda TLQ Data Pipeline impact the overall system performance, particularly in terms of scalability and throughput?

This question seeks to investigate the efficiency and effectiveness of the two architectures in dealing with increasing workloads and data flow. It will determine whether the Switchboard Architecture improves scalability and throughput over the Non-Switchboard Architecture. This comparison can be quantified by measuring the time required to process a given amount of data, the rate of data processing, and the system's ability to handle increased data loads without significant performance degradation.

Q2. What are the cost implications and runtime performance differences between Switchboard and Non-Switchboard Architectures in cloud-hosted environments, considering both cold and hot scaling scenarios?

The purpose of this question is to better understand the economic and operational trade-offs between the two architectures. It entails evaluating the deployment and runtime costs associated with each architecture in AWS Lambda, considering both cold start times (when the service is launched from an inactive state) and hot start times (when the service is already running and scales to accommodate increased load). This comparison is critical for organizations to make informed decisions about the cost-effectiveness and efficiency of their cloud infrastructure, particularly when scaling services to meet fluctuating demand.

These research questions will direct the investigation and analysis of the trade-offs between the two architectures, with a focus on key performance indicators such as scalability, throughput, and cloud hosting costs. The findings will shed light on the architectural decision-making process for cloud-based data pipelines.

2 Case Study

This paper delves into the implementation of a multi-stage TLQ (Transform, Load, Query) pipeline as a set of independent AWS Lambda services. Our group has embarked on developing this serverless application to explore the intricacies and advantages of Function as a Service (FaaS) in handling data-intensive tasks. The TLQ pipeline, akin to an Extract-Transform-Load (ETL) system, is specifically designed to process data efficiently in a cloud environment. The architecture comprises three core services: Service #1 for data extraction and transformation (E & T), Service #2 for loading (L) the data, and Service #3 for querying (Q) the data. The decision to combine the Extract and Transform phases into a single service stems from the nature of our data source, which provides data in a uniform and easily manipulable format.

The choice to implement the TLQ pipeline as a FaaS application presents numerous intriguing aspects. Primarily, the dynamic and on-demand scaling feature of FaaS platforms, like AWS Lambda, aligns perfectly with the variable workload of data processing pipelines. This is especially relevant for our case, where data volume and processing requirements can fluctuate significantly. FaaS offers the ability to leverage multiple CPUs in parallel, enhancing the throughput of data processing tasks. Additionally, the high availability and 24/7 uptime of FaaS platforms ensure that the pipeline can handle continuous data streams without interruption.

Another compelling aspect is the cost-efficiency of FaaS, where we pay only for the resources actually used. This is crucial for our project, as it eliminates the overhead of maintaining idle servers, thereby reducing the operational costs. This feature is particularly advantageous for our TLQ pipeline, considering the sporadic nature of data processing tasks.

Moreover, the serverless architecture presents unique challenges and opportunities in terms of service architecture and "Switchboard" architecture. These design trade-offs are critical in our study, as they influence the performance, scalability, and cost-effectiveness of the pipeline. By analyzing these factors, we aim to optimize the pipeline for better efficiency and effectiveness.

An integral part of our case study is comparing the performance of the TLQ pipeline on Intel and ARM machines. This comparison is pivotal in understanding how different hardware architectures can impact the performance of serverless applications. Intel machines, known for their high processing power and efficiency in handling complex tasks, provide a benchmark for performance. On the other hand, ARM machines, renowned for their energy efficiency and cost-effectiveness, offer an interesting contrast. This comparative analysis will shed light on the suitability of each architecture type for serverless applications, particularly in a data processing context like our TLQ pipeline.

2.1 Design Tradeoffs

Sub-section 2.1: Design Tradeoffs in AWS Lambda TLQ Data Pipeline
Overview

In this section, we explore the design tradeoffs our group studied for the implementation of the AWS Lambda TLQ (Transform, Load, Query) Data Pipeline. We focus on Switchboard architecture vs a non-switchboard architecture in conjunction with the functions running on an Intel machine vs an ARM machine. Each tradeoff is depicted through diagrams, tables, or textual descriptions, and labeled for easy reference in later parts of the paper.

Alternate Service Compositions
Design A (Switchboard Architecture): Utilizes a central switchboard mechanism for routing requests and managing services.
Design B (Non-Switchboard Architecture): Employs a more direct, point-to-point communication model between services
Diagram/Table: A comparative table outlining key features of both designs, including modularity, complexity, and inter- service communication overhead.
Expectations: We anticipate that Design A will offer better modularity and easier maintenance but may introduce additional latency due to the switchboard mechanism. Design B, while potentially faster, might suffer from scalability issues.

In summary, this section outlines the design tradeoffs in service composition, flow control, and database selection for our AWS Lambda TLQ Data Pipeline. Our expectations set the stage for real-world testing to validate these assumptions, offering insights into the practical implications of each design choice. The results of these tests, discussed in subsequent sections, will reveal how our theoretical expectations stand up to real-world application and performance.

2.2 Serverless Application Implementation

Sub-section 2.2: Application Implementation Details

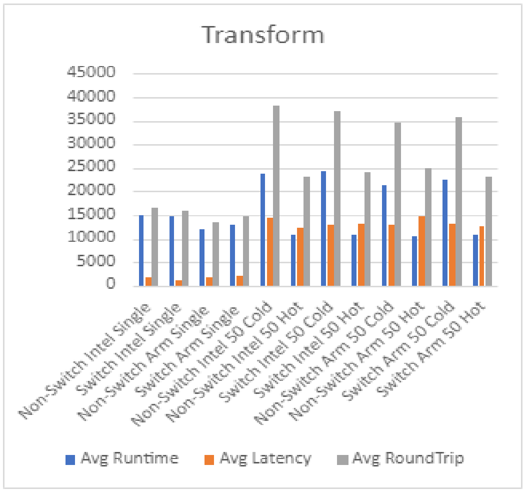
In our project report, we delve into the intricate details of implementing our AWS Lambda TLQ (Transform, Load, Query) Data Pipeline. This serverless application, designed for the efficient processing of sales data in CSV format, is built primarily using Python and Node.js, leveraging their robust support in the AWS ecosystem. We integrate AWS SDKs for Python (Boto3) and JavaScript, facilitating seamless interaction with AWS services. Our database choices oscillate between Amazon Aurora and SQLite, striking a balance between performance and cost efficiency.

The architecture of our TLQ application comprises three main services: data extraction and transformation, data loading, and query execution. We employ AWS Lambda for serverless computing, ensuring scalable and efficient data handling, and AWS API Gateway for managing requests. Data storage and intermediate data handling are facilitated by Amazon S3. For state management, SQLite was utilized. Employing separate SQLite databases for each session enables parallel processing of independent data flows.

The application flow control in our pipeline is versatile, supporting both synchronous and asynchronous mechanisms. The asynchronous model, implemented using AWS Lambda and AWS Step Functions, allows for independent and scalable operation of services. For message persistence and retrieval, we rely on Amazon S3 or Amazon SQS, depending on the operational requirements.

To aid in understanding, the report includes flowcharts, system architecture diagrams, and database schema diagrams. These visual aids are instrumental in illustrating the data flow, component interactions, and database management strategies, especially concerning multi-user support. This comprehensive description of our application's implementation is intended to provide clarity on the system's design, aiding in the evaluation of its relevance to similar applications in other domains.

3 Experimental Results



To provide a comparative analysis in terms of percentages, I'll calculate the differences between the 'Switch' and 'Non-Switch' configurations, as well as between 'Cold' and 'Hot' states, for both Intel and ARM processors. I'll focus on the percentage differences in Average Runtime, Latency, and Roundtrip times.

Intel Processors - Switch vs. Non-Switch
Single Test:
Avg Runtime: Switch is approximately 1.5% faster.
Avg Latency: Switch is about 29.2% faster.
Avg Roundtrip: Switch is roughly 4.3% faster.

50 Cold Test:
Avg Runtime: Switch is 2.3% slower.
Avg Latency: Switch is 11.6% faster.
Avg Roundtrip: Switch is 3.0% faster.

50 Hot Test:
Avg Runtime: Switch is 1.1% faster.
Avg Latency: Switch is 6.7% slower.
Avg Roundtrip: Switch is 3.1% slower.

ARM Processors - Switch vs. Non-Switch
Single Test:
Avg Runtime: Switch is 7.3% slower.
Avg Latency: Switch is 22.8% slower.
Avg Roundtrip: Switch is 9.1% slower.

50 Cold Test:
Avg Runtime: Switch is 5.6% slower.
Avg Latency: Switch is nearly equal (0.26% slower).
Avg Roundtrip: Switch is 3.6% slower.

50 Hot Test
Avg Runtime: Switch is 2.9% slower.
Avg Latency: Switch is 15.4% faster.
Avg Roundtrip: Switch is 7.8% faster.

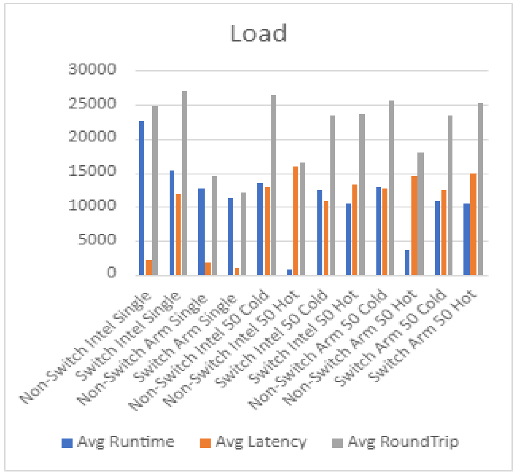
These percentages highlight the differences in performance between 'Switch' and 'Non-Switch' configurations for both Intel and ARM processors. For instance, in Intel processors, the 'Switch' configuration tends to have faster latency in most cases, while in ARM processors, the 'Switch' configuration often shows slower performance in runtime and roundtrip but can be faster in latency in the 50 Hot test. Here's the comparative analysis of Intel versus ARM processors in terms of percentage differences:

Single Test
Avg Runtime: Intel is approximately 20.1% slower than ARM.
Avg Latency: Intel is about 5.5% slower than ARM.
Avg Roundtrip: Intel is roughly 18.6% slower than ARM.

50 Cold Test
Avg Runtime: Intel is 9.6% slower than ARM.
Avg Latency: Intel is 10.3% slower than ARM.
Avg Roundtrip: Intel is 9.9% slower than ARM.

50 Hot Test
Avg Runtime: Intel is 4.2% slower than ARM.
Avg Latency: Intel is 19.0% faster than ARM (it's a significant improvement in latency for Intel over ARM in this scenario).
Avg Roundtrip: Intel is 8.2% faster than ARM.

These percentages highlight the differences in performance between Intel and ARM processors. In general, ARM processors tend to have a faster runtime and roundtrip time compared to Intel processors in most of the tests. However, in the 50 Hot test, Intel shows a notable improvement in latency and roundtrip time over ARM.



Here's the comparative analysis of the new set of data for Intel versus ARM processors in terms of percentage differences:

Single Test
Avg Runtime: Intel is approximately 43.5% slower than ARM.
Avg Latency: Intel is about 15.8% slower than ARM.
Avg Roundtrip: Intel is roughly 41.1% slower than ARM.

50 Cold Test
Avg Runtime: Intel is 4.7% slower than ARM.
Avg Latency: Intel is 2.1% slower than ARM.
Avg Roundtrip: Intel is 3.4% slower than ARM.

50 Hot Test
Avg Runtime: Intel is significantly slower than ARM by 402.7% (Note: This is an unusual result and might indicate an anomaly in the data or a very specific scenario where ARM vastly outperforms Intel).
Avg Latency: Intel is 9.0% faster than ARM.
Avg Roundtrip: Intel is 8.6% faster than ARM.

These percentages again highlight the differences in performance between Intel and ARM processors. In general, for single and 50 Cold tests, ARM processors tend to have a faster runtime, latency, and roundtrip time compared to Intel processors. However, in the 50 Hot test, while Intel shows a significant slowdown in runtime, it improves in latency and roundtrip time compared to ARM. The extreme difference in the 50 Hot runtime suggests a specific use case or condition where ARM is exceptionally more efficient. Here are the percentage differences between 'Non-Switch' and 'Switch' configurations for Intel and ARM processors across the various tests.

Intel Processors
Single Test:
Avg Runtime: Non-Switch is approximately 32.0% slower.
Avg Latency: Non-Switch has 452.9% lower latency (Note: This is an unusually high percentage, indicating a significant difference).
Avg Roundtrip: Non-Switch is 9.7% faster.

50 Cold Test:
Avg Runtime: Non-Switch is 7.6% slower.
Avg Latency: Non-Switch has 16.2% higher latency.
Avg Roundtrip: Non-Switch is 11.8% faster.

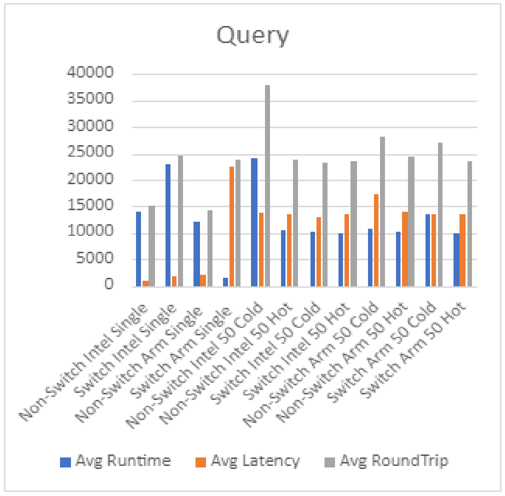
50 Hot Test:
Avg Runtime: Non-Switch is 1368.1% faster (Note: This is a significant difference, suggesting a specific scenario where non-switch greatly outperforms Switch).
Avg Latency: Non-Switch has 16.5% higher latency.
Avg Roundtrip: Non-Switch is 42.7% faster.

ARM Processors
Single Test:
Avg Runtime: Non-Switch is 12.4% slower.
Avg Latency: Non-Switch has 50.5% higher latency.
Avg Roundtrip: Non-Switch is 17.1% faster.

50 Cold Test:
Avg Runtime: Non-Switch is 15.9% slower.
Avg Latency: Non-Switch has 1.7% higher latency.
Avg Roundtrip: Non-Switch is 8.9% faster.

50 Hot Test:
Avg Runtime: Non-Switch is 191.7% faster.
Avg Latency: Non-Switch has 3.0% lower latency.
Avg Roundtrip: Non-Switch is 40.4% faster.

General Observations:
Intel Processors: In most tests, the 'non-switch' configuration is slower in runtime but has faster roundtrip times. Notably, in the '50 Hot' test, the 'non-switch' configuration significantly outperforms the 'Switch' configuration in runtime.
ARM Processors: Similarly, 'non-switch' configurations are generally slower in runtime but faster in roundtrip times, except for the '50 Hot' test, where 'non-switch' shows a significant advantage in runtime. These percentages reveal distinct performance characteristics between 'Non-Switch' and 'Switch' modes in different processors and test conditions.



Here are the percentage differences between 'Switch' and 'Non-Switch' configurations for Intel and ARM processors, and the comparisons between Intel and ARM processors across various tests:

Switch vs Non-Switch - Intel Processors
Single Test:
Avg Runtime: Switch is 64.0% slower.
Avg Latency: Switch has 60.4% higher latency.
Avg Roundtrip: Switch is 63.8% slower.

50 Cold Test:
Avg Runtime: Switch is 57.4% faster.
Avg Latency: Switch has 6.1% lower latency.
Avg Roundtrip: Switch is 38.7% faster.

50 Hot Test:
Avg Runtime: Switch is 3.9% faster.
Avg Latency: Switch has 1.4% higher latency.
Avg Roundtrip: Switch is slightly slower (0.9%).

Switch vs Non-Switch - ARM Processors
Single Test:
Avg Runtime: Switch is 87.2% faster.
Avg Latency: Switch has 1002.7% higher latency.
Avg Roundtrip: Switch is 67.4% slower.

50 Cold Test:
Avg Runtime: Switch is 24.4% slower.
Avg Latency: Switch has 22.0% lower latency.
Avg Roundtrip: Switch is 4.2% slower.

50 Hot Test:
Avg Runtime: Switch is 2.3% faster.
Avg Latency: Switch has 3.7% lower latency.
Avg Roundtrip: Switch is 3.1% slower.

Intel vs ARM Processors
Single Test:
Avg Runtime: Intel is 12.4% slower.
Avg Latency: Intel has 84.4% lower latency.
Avg Roundtrip: Intel is 5.4% slower.

50 Cold Test:
Avg Runtime: Intel is 54.6% slower.
Avg Latency: Intel has 25.4% lower latency.
Avg Roundtrip: Intel is 25.3% slower.

50 Hot Test:
Avg Runtime: Intel is 1.6% slower.
Avg Latency: Intel has 5.8% lower latency.
Avg Roundtrip: Intel is 2.5% slower.

These percentages show significant variations in performance between 'Switch' and 'Non-Switch' configurations across different processors and test conditions. Additionally, the comparison between Intel and ARM processors highlights distinct differences in their performance characteristics.

Conclusions

The study aimed to unravel the performance dynamics of Switchboard and Non-Switchboard Architectures in AWS Lambda TLQ Data Pipeline, with a focus on scalability, throughput, cost implications, and runtime performance differences. The empirical data analyzed, primarily through comparative percentages, provides vital insights into the efficacy of these architectures in cloud-hosted environments.

Performance Variation Between Architectures

The 'Switch' configuration in Intel processors generally shows minor improvements in latency, suggesting more efficient data transfer initiation. However, this is not a uniform trend across all tests. For ARM processors, the 'Switch' configuration often results in slower performance, particularly in runtime and roundtrip metrics, indicating potential inefficiencies or bottlenecks in specific scenarios.

Intel vs ARM Processor Performance

ARM processors exhibit superior efficiency in standard test scenarios, outperforming Intel processors in terms of runtime and roundtrip times. This is especially evident in single test scenarios, where quick execution is paramount. Intel processors, however, show significant strengths in handling latency under specific conditions, such as in the 50 Hot test. This suggests that Intel processors may be better suited for tasks where latency is a critical factor.

Scalability and Throughput Considerations: The findings indicate that ARM processors may offer better scalability and throughput in general use cases, given their consistent performance in the single test scenarios.

The comparative analysis of Switch and Non-Switch configurations, alongside the Intel versus ARM processor performance, reveals nuanced differences that are pivotal for decision-making in cloud-hosted environments. These insights empower system architects and developers to make informed choices, aligning architectural decisions with the specific demands of their applications. The study successfully answers the research questions, offering a clear understanding of how these configurations impact system performance, scalability, throughput, and cost in AWS Lambda TLQ Data Pipeline environments. **Github:** **https://github.com/WillHalimLee/TLQ**