Amazon Advertising Data: An Automated AWS Pipeline

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Johnathan Middleton

Spring, 2024

Technical Project

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Briana Morrison, Department of Computer Science

Amazon Advertising Data: An Automated AWS Pipeline

CS4991 Capstone Report, 2023

Johnathan Middleton
Computer Science
The University of Virginia

School of Engineering and Applied Science

Charlottesville, Virginia USA

jpm6qjz@virginia.edu

ABSTRACT
Amazon Sponsored Brands, a sub-division of Amazon's advertisement department, decided to optimize a weekly business review sheet by creating an automated pipeline for pushed advertisements. I utilized various Amazon Web Services (AWS) services to create a periodically-run system that would efficiently gather, aggregate and report upon the metrics, generating the necessary reports in a timely and predictable manner. To orchestrate the pipeline, I used AWS's Cloud Development Kit (CDK) within which I used AWS services such as Elastic MapReduce (EMR) and Step Functions to implement the individual steps. The pipeline reduced significant intermediate work required to compile the final data. The future of the pipeline consists of verifying the unit tests, and next steps include increasing the input data size when new data sources are added.

1. INTRODUCTION

Consider a weekly business report, hereafter referred to as a WBR, consisting of various advertising related business metrics such as CPM (cost-per-thousand impressions) or CPC (cost-per-click). These metrics are vital to understanding the effectiveness of marketing strategies deployed by a corporate entity. This report should be three things: timely, comprehensive, and accurate. In practice, the generation of such reports involves huge swaths of data that take part in a deeply layered pipeline involving aggregation and analysis.

To ensure the timely nature of the WBR pipeline, one must consider its scheduling and efficiency. Data may be collected at a unique rate, independent of the WBR's expected date, so it is necessary to plan for this. Additionally, one must establish an agreed-upon turnaround time that separates the collection of data and its inclusion in the WBR. For comprehensiveness, it is vital to include all relevant data points. This means the schema in which advertising data is collected must not be affected in a way not cohesive with the WBR. It is also important to prepare for edge cases in collection patterns, such as unusually large amounts of data during a busy holiday. Finally, the pipeline must be accurate. For accuracy, various functions and calculated values contributed to the final WBR must be preserved through the processes of the pipeline, and modifications made in various steps must not affect the expected resulting calculations.

2. RELATED WORKS

There is significant research being done in the field of NoSQL (Not Only SQL), which consists of "a wide scope of database propels that were made in light of the limit of a significant volume of customer data" (Monika & Shrivastava, 2020). These systems, which boast high versatility and computational times, are limited in their usefulness when handling disproportionately large amounts of data. The heavily documented patterns in a relational database management system (RDBMS) are better suited for advertising data in this context of a large corporation. Monika and Shrivastava detail this paradigm and explore the emerging patterns of NoSQL. All research was done in the lens of an RDBMS, but it is worth mentioning the viability of a NoSQL design.

When implementing a large-scale data pipeline, there are many "Challenges and Opportunities" (Munappy et al., 2020). We can turn to Munappy to better understand the design decisions to be made when planning a pipeline, including performance, ease of use, validity, and scalability. Munappy helps us to better understand how pipelines play a role in the research done by these companies.

Rettenmeier and Bogdanov (2021) discuss the elements of an event-driven architecture, a pattern for designing a data aggregation pipeline. The pipeline in this context follows a modified version of this pattern, and comparing the two provides a valuable insight into the efficiency losses and gains of each. High throughput of an event-driven pipeline is a necessity, and following a design pattern that enforces a consistent rate of action paired with a capacity for scalability is an effective combination.

3. PROJECT DESIGN

The design of the pipeline was complex in nature, and thus was separated into two parts – a high-level one and a low-level one. The high level was meant to provide a bird's-eye view of the pipeline and how it integrated into the team's cloud environment, and the low-level was meant to provide specific implementation details before the implementation could begin.

3.1 Pipeline High-Level Design

The high-level design was an over-arching look at how the pipeline's infrastructure would operate, and laid the groundwork for the low-level design.

3.1.1 Components

Involved in the high-level design were several AWS services to be used to orchestrate the various processes at work. The sum of these parts was described using AWS Cloud Development Kit (CDK). This provided a reproducible way to test the functionality of the pipeline. The EventBridge service is used as a scheduler for the processes involved and is set up to run at specific intervals throughout the day. The scheduler takes in as an input a target, and when the scheduler invokes this target, executing its functionality, arguments can be passed in. In this way, different data sets were operated upon. The target defined was a Step Function instance, which would allow specific tasks to be run in a deterministic fashion, ensuring one element of the pipeline completed successfully before invoking the next. The individual steps within the step function described the data aggregation process.

For data storage, S3 buckets were utilized as object storage for the various outputs of the step function. The tree structure of S3 buckets was the ideal choice for this format of storage. For the computational side of things, Amazon Elastic MapReduce, or Amazon EMR was used as the big data platform. Apache Spark was used as the accompanying engine, as based on the operations involved in the data aggregation process, this was best suited.

3.1.2 Scheduling

A critical component of a pipeline which must be timely is the order and rate at which it performs its tasks. The execution of the previous pipeline took roughly eight hours to complete an entire run. This run included all the country codes that were to be used, as well as all the product groups in one huge run.

One goal of the proposed pipeline was to modularize this. To do this, I designed a plan to execute the various country codes and product groups' advertising metrics in parallel with each other, using EMR clusters, which are dynamically allocated servers designed to be used for on-the-fly computational workloads. The schedule involved a start time of 8 A.M. E.S.T, when EMR clusters would be spun up in parallel to individually handle the product groups. Then spread out through the day would be the country codes. This allowed for the resulting daily business reports to be outputted at the times appropriate to each country codes time zone.

This scheduling accounts solely for the daily accumulation of data, however. To properly generate the WBRs, some weekly scheduling is necessary. Advertising data is cumulative so the data points of a specific day may be related to that of another, as a customer receives an ad one day of the week and purchases a product another. Due to this factor, simply aggregating data daily is not enough to generate the relevant metrics.

To accomplish this, I included another scheduling function in the EventBridge instance, that would run a large-scale weekly aggregation on the data collected from the previous week. This allowed for consistent generation of the WBRs and removed a costly dependency, as the daily aggregation was now unaffected by the weekly aggregation, so a failure in the weekly process would not result in a loss of data.

3.2 Pipeline Low-Level Design

The low-level design was proposed after the high-level design was finalized, and was to provide a detailed guide for implementation.

3.2.1 Data Specifications

We started with three main sources of data: impressions, auctions, and conversions. Impressions included data points relevant to advertisements pushed to the consumer. This included metrics such as the number of advertisements shown to a particular customer as well as a retrieval rate from advertisement sources. Auction data consisted of information relevant to the auction for each advertisement spot, in which bidding companies that desire to have their advertisement shown compete. This would include the density of the auction and amounts bid. Finally, conversions were the result of the advertising cycle, where successful conversions of advertisements were recorded. This table was smaller in size and was filled with data up to a week after its corresponding data in impressions or auctions data sources.

3.2.2 Flow of Data

The collected data had to follow a precise schema to be accurate. The step function began by joining the tables of impressions and auctions. This paired up the individual impressions by the relevant auction that occurred to generate that impression.

Once the step function verified this phase was complete, it would spin up another and begin joining the resulting auction records table with the conversions table. As the conversions table was not generated until a week after its counterpart, the impressions table, this process had to be delayed by a week. Following successful joins, we were able to view the metrics of winning bid, the expected and real click through rate of the impression, as well as the reserve price, which varied based on the weight of the impression as well as the auction information. The step function then executed the aggregation steps.

This generated metrics based on the winners and losers of an auction, giving the following values: ROAS, return on ad spending; RPOM, a measure of revenue; and monetization rate, essentially the effectiveness of an advertisement. Once these aggregated auction records were generated and verified, the daily business report could be created. The aggregated auction records were converted into the correct format by another EMR cluster, and the resulting business report was written to an S3 bucket.

4. RESULTS

After the implementation of the redesigned pipeline, the product was quickly able to be moved into a production environment. As its predecessor was still operational, it was quite straightforward to verify the redesign's important qualities: timeliness, comprehensiveness, and accuracy. For the efficiency of the aggregation process, the execution of the redesign had parallelized and modularized the computation on country codes and product groups. This brought down individual execution times significantly.

Based on the sum of the independent runs of the pipeline compared to the monolithic completion time of the previous, the redesign achieved a sum time of five hours and 58 minutes, compared to a time of eight hours and 27 minutes. This test was run on an identical data set of impressions, auctions, and conversions, and constitutes a 29.3% decrease in completion time. Verifying the comprehensiveness of the redesign involved an analysis of the extensibility built-in to the design. Through support for relevant data sets and parameter matching, the redesign was able to handle more complex combinations of aggregation patterns than its predecessor. For accuracy, we compiled an individual weekly business report using each of the pipelines, and analyzed the percent error in each of the computed values. We were able to achieve below a 0.01% percent error in all of the relevant advertising metrics.

5. CONCLUSION

Construction of a data aggregation pipeline is a broad and time-intensive task; however by following precedent and adhering to the ideas of timeliness, comprehensiveness, and accuracy, one can produce a valuable output that will prove useful and have longevity. The most important aspects of the pipeline were the scheduling functionality, which allowed for parallelization of tasks operating upon large data payloads, and the precise capture of the data flow, which solidified a consistent schema to be built upon. I gained through the planning and implementation of the pipeline skill in planning the instrumentation of data aggregation, as well as experience conforming to specific data specifications.

6. FUTURE WORK

The pipeline was implemented and deployed to cloud servers to replace its predecessor, and it remains in operation as the supplier for the WBR. The future oriented design allows for additional data sources to be easily added in the future, and the modularity enabled by use of Amazon's CDK promotes adoption from alternate teams. A possible improvement on the pipeline is a decreased dependency on a particular data flow, allowing for more extensibility. Another is fault tolerance, which would handle edge cases in the data sets, such as invalid or entirely missing data points. While the pipeline operates in a specifically defined manner pertaining to advertising metrics, the design components such as scheduling and rigid schemas could be implemented in other fields, bringing with it the added performance.

REFERENCES

Monika, & Shrivastava, V. (2020). An Approach to Improved MapReduce and Aggregation Pipeline Utilizing NoSQL Technologies. *Lecture Notes in Networks and Systems*. Retrieved November 21, 2023, from https://doi.org/10.1007/978-981-15-7561-7_15

Munappy, A. R., Bosch, J., & Olsson, H. H. (2020). Data Pipeline Management in Practice: Challenges and Opportunities. *Product-Focused Software Process Improvement*, 168–184. Retrieved October 18, 2023, from https://doi.org/10.1007/978-3-030-64148-1_11

Rettenmeier, L., & Bogdanov, K. (2021, November 1). *Build a near real-time data aggregation pipeline using a serverless, event-driven architecture | AWS Database Blog*. Https://Aws.amazon.com/Blogs/Database/Build-a-Near-Real-Time-Data-Aggregation-Pipeline-Using-a-Serverless-Event-Driven-Architecture/. Retrieved October 18, 2023, from https://aws.amazon.com/blogs/database/build-a-near-real-time-data-aggregation-pipeline-using-a-serverless-event-driven-architecture/