Cloud Migration and GraphQL Implementation to Improve Cost Measures

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

Shriman Selvamani

Fall 2022

Aaron Bloomfield, Department of Computer Science

Cloud Migration and GraphQL Implementation to Improve Cost Measures

CS4991 Capstone Report, 2023

Shriman Selvamani
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
ss5qt@virginia.edu

ABSTRACT
The existing data infrastructure that ran on-premises systems at CoStar Group posed substantial challenges in terms of cost, maintenance, and scalability. Additionally, the middleware layer, acting as the Backend for Frontend, suffered from network latency issues due to an excess of information being carried in API requests. To address these concerns during my internship, I implemented a comprehensive solution that involved migrating all data to the cloud on AWS and transitioning the middleware to GraphQL. I executed data migration using PowerShell scripts and conducted in the development, test, and production environments. The project enabled significant cost savings through cloud utilization, and substantial improvements in network latency due to the more efficient API load. Future work may encompass extending the migration process to other components of the CoStar Group website, thereby maximizing the benefits of a cloud-based infrastructure throughout the product.

1. INTRODUCTION

In the rapidly evolving landscape of data management and infrastructure, organizations are constantly seeking ways to enhance efficiency, reduce operational costs, and improve the reliability of their systems. Traditional on-premise data centers have long been a staple for organizations, but their limitations in terms of scalability, maintenance costs, and susceptibility to single points of failure have become increasingly apparent. One solution is Amazon Web Services (AWS) DynamoDB database, offering organizations a user-friendly, highly scalable, and fully managed database platform that simplifies data storage, retrieval, and management while upholding standards of reliability and security. This shift in data management has compelled various industries, including the commercial real estate sector, to explore and implement cloud-based alternatives.

My project dealt with data infrastructure. Specifically, it explored a migration process undertaken at CoStar Group, a prominent player in the commercial real estate sector, as they transitioned from on-premise data centers to the AWS cloud. A critical aspect of this migration involved leveraging the Modify Table Layout feature on the CoStar Suite website, which is a comprehensive real estate software and data service. This feature allows users to customize the properties of commercial listings for viewing in a table format, helping users see only what they want to see.

2. RELATED WORKS

The migration performed at CoStar is not new to the technology industry. Ever since cloud computing was launched, the popularity of leveraging the cloud was very high. Companies such as Capital One and Ocado Technology adopted and migrated to DynamoDB to reap its benefits.

The adoption of Amazon Web Services DynamoDB by Capital One was a strategic move driven by the need for a highly resilient and low-maintenance database solution. DynamoDB stood out to Capital One for its cross-region active-active capabilities and high performance, even though it sacrificed some of the data access flexibility inherent in SQL-based databases. The migration to DynamoDB yielded substantial benefits for Capital One, with a remarkable 99% reduction in application failover time. The elimination of regional failovers and AWS's data replication capabilities played a crucial role in achieving this. Additionally, the query performance remained on par or improved, despite some initial database design optimizations. While the migration presented a few challenges, the overall outcome justified the effort, significantly enhancing failover speed and simplifying the complexity of the system. This experience underscores the advantages of leveraging managed database services like DynamoDB to streamline operations and improve system resilience (Brown, 2021).

Much like Capital One's experience, Ocado Technology highlights the importance of understanding application access patterns and effectively utilizing DynamoDB's features, including partition and index overloading, to optimize database performance. While this approach may lead to tables that resemble computer language more than human-readable RDBMS tables and might reduce ad-hoc querying flexibility, it aligns with the trade-off between flexibility and the exceptional characteristics of DynamoDB, including speed, reliability, consistency, and its ability to scale seamlessly to handle any workload (Muc, 2022).

3. PROJECT DESIGN

The project at CoStar Group encompassed two critical components: the migration of the database to Amazon Web Services (AWS) and the transition of the middleware layer to GraphQL.

3.1 System Architecture Changes

To restate, there were two requirements of this project: 1) Changing the middleware layer from the current backend for frontend (BFF) layer to GraphQL; and 2) Moving the backend SQL Server database called PDS, which was hosted on premises, to the AWS Cloud Dynamo DB to a database called CUE (Central User Entity) service. These proposed changes can be seen in the design plan as shown in Figure 1.

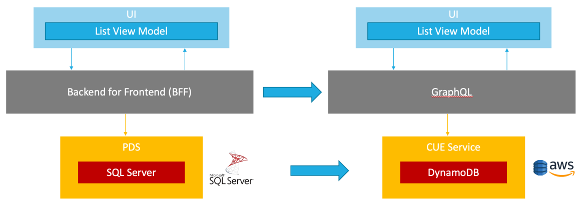

Figure 1. Proposed Architecture Plan

Implementation and design of this solution required a multifaceted approach, with data migration and middleware transformation happening concurrently. This process was carefully designed so the changes would be more efficient and still allow the features to function effectively.

3.2 Data Migration

Data migration required all the data to be moved from the on-premises database to the cloud flawlessly, with no data lost or destroyed in the process. Transformation of data to fit the new database was done during migration. The migration was conducted using SQL scripts to access the specific databases located on Premises, which was an

SQL server database named PDS. The migration was conducted in order of dev, test, and production environments. To validate that the actions were expected and that there were no issues prior to handling vital production data.

3.3 GraphQL

Following the successful data migration, client implementation had to be done to define the new GraphQL API and how the API would be presenting information from the new cloud database server. For this stage, we loaded columns from CUE database on AWS Cloud on page load and integrated them with existing data flow. We then saved the modify table layout modal into CUE as each grid should have its own saved layout. We also had a feature toggle flag, used during development to enable testing of the new and old functionalities.

Next, we designed the API, as shown in Figure 2. The design process began with creating the API architecture, which was later integrated GraphQL to interact seamlessly with React Hooks. We created the API architecture by combining two existing endpoints, one for serving most grids in CoStar Suite and the other for public records. We then analyzed the needs of the product to determine the features, so one endpoint could store Column ID, layoutID, type, and Search by product, grid, and country.

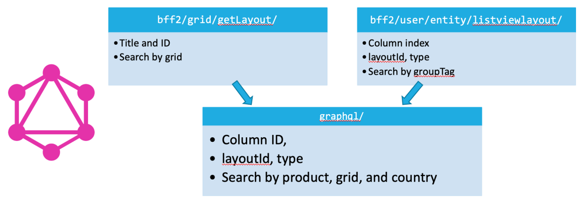

Figure 2. API Design

GraphQL in Suite uses React Hooks, which can only be used with functional components. Because many of the grid components are class components, the solution was to create a custom React Hook that handles all the implementation details, and create a helper component for each product grid. Each grid has custom actions to transform the data once fetched.

4. RESULTS

The Modify Table Layout feature data migration and middleware change improved performance, cost, complexity, modernization and efficiency; and these were very valuable improvements brought to CoStar Suite. Specifically, with the data migrated from the on-premises data center to Amazon Web Services DynamoDB, we were able to receive benefits of the cloud such as scalability, cost saving, and utilization of a fully-managed service. Similarly, the GraphQL implementation helps improve efficiency by utilizing a single endpoint that improves speed and reduces the number of requests to endpoints required.

5. CONCLUSION

The project carried out at CoStar Group focused on two essential elements: transitioning the database to Amazon Web Services (AWS) and shifting the middleware to a GraphQL-based system. The need for migrating to AWS came from the limitations of the existing on-premises data infrastructure. These challenges included high costs, intensive maintenance requirements, and scalability issues. I executed this migration using PowerShell scripts across different environments - development, test, and production. This move to AWS, particularly to DynamoDB, allowed us to benefit from the cloud's scalability, cost efficiency, and enhanced maintenance capabilities.

Simultaneously, I addressed the inefficiencies in our existing API setup by transitioning our middleware from a Backend for Frontend (BFF) layer to GraphQL. This change was aimed at overcoming network latency issues and improving API efficiency. The implementation of GraphQL involved designing a new API that was integrated with our AWS cloud database and ensuring its seamless operation.

The results of these implementations were significant. The migration to AWS DynamoDB provided us with improved scalability, efficiency, and cost savings, while the adoption of GraphQL led to better network latency and enhanced API load efficiency. This project at CoStar Group shows how cloud migration and middleware optimization can substantially improve the performance, cost-effectiveness and scalability of data infrastructure in large organizations. The successful deployment of AWS and GraphQL not only resolved the existing challenges but also laid a strong foundation for future technological enhancements and potential expansions in CoStar Groups digital infrastructure.

6. FUTURE WORK

Several avenues can be explored to further enhance and expand upon the current project's achievements. First, a comprehensive evaluation of the long-term performance and cost benefits of the AWS and GraphQL implementations in different operational scenarios at CoStar Group is essential. This would involve monitoring the system under varying loads and use cases to identify areas for optimization and to ensure that the infrastructure scales effectively with the company's growth. Additionally, exploring the integration of advanced AWS services like AWS Lambda and Amazon Elastic Container Service (ECS) could provide further improvements in efficiency and scalability.

Another promising area for future development is the extension of GraphQL capabilities. Implementing advanced GraphQL features like subscriptions for real-time data updates and incorporating more sophisticated query optimization techniques could significantly enhance the API's performance and usability. Furthermore, the successful migration and middleware transition at CoStar Group opens possibilities for replicating this model in other divisions of the company or even in different sub-organizations facing similar challenges. Sharing the learnings and methodologies from this project could provide valuable insights for others looking to undertake similar digital transformation initiatives.

REFERENCES

Muc, T. (2022, January 18). *Using Amazon DynamoDB to store data at scale*. Medium. Retrieved September 28, 2023, from https://medium.com/ocadotechnology/using-amazon-dynamodb-to-store-data-at-scale-1496895b9945

Brown, K. J. (2021, February 10). *Moving to DynamoDB to increase application resiliency*. Capital One. Retrieved September 28, 2023, from https://www.capitalone.com/tech/software-engineering/comparing-dynamodb-and-aurora-global-database-and-aurora-multi-master/