Streamlining Cvent's Data Fetching Process

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

Moeez Sohail

Fall, 2022

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

Rosanne Vrugtman, Department of Computer Science

Streamlining Cvent's Data Fetching Process

CS4991 Capstone Report, 2022

Moeez Sohail
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
ms7gf@virginia.edu

ABSTRACT
Cvent, a Virginia-based software as a service company, found that its Instant Book platform inefficiently obtains data from external sources. As a result, I replaced existing data access modules with Apollo data sources, modern classes that increase efficiency when obtaining information from a specific source. I completed a full refactor where I removed the previous accessors from the software and updated dependent code to use the new data sources. By upgrading the code base to use Apollo data sources, data fetching from REST APIs through GraphQL queries resulted in fewer errors when completing operations. Additionally, the platform achieved quicker results due to enhanced caching. However, some of the existing data access modules still need to be refactored to ensure that the Instant Book platform has the best possible overall performance.

1. INTRODUCTION

Cvent is a software company that provides software solutions to event planners for virtual, hybrid, and onsite events. One platform they provide is Instant Book which allows small businesses to find and book venues. Users can specify if they need meeting rooms or guest rooms, when they will need the venue, how many people, and what setup they need. Instant Book uses these search filters to find matching venues and allow users to instantly book. When searching, Instant Book fetches data from a variety of different data providers to increase the number of potential venues. These GraphQL queries to REST APIs play an essential role in the overall performance of the platform since it is mostly used to search for and find venues. After noticing a decrease in performance in Instant Book due to slow query processing, my team and I realized the need to optimize the queries to retain platform users and improve user experience.

2. RELATED WORKS

Cochrane and Debrunner (2022) outline different methods that can be used to optimize GraphQL queries. Some of these methods include deduplication and caching. Deduplication is a technique that prevents duplicate GraphQL queries from running in the server [1]. Caching is a technique that keeps local copies of frequently accessed data so that it can be obtained faster instead of having to go through databases [1]. The Apollo data sources solution I implemented utilizes deduplication and caching to improve performance of GraphQL queries to external REST APIs.

In an article by Sands-Ramshaw (2020), the author explains how to use Apollo data sources and how they can be used to help optimize GraphQL queries. By default, Apollo data sources utilize an in-memory cache to store results of previous operations. For REST data sources specifically, caching

is based on the HTTP response's Cache-Control header and lifecycle methods [2]. I utilized this REST data source parent class to solve the Instant Book platform's inefficiency issues.

3. PROCESS DESIGN

When it comes to the design of the project, a complete refactor must be completed. The existing data accessors will be replaced with optimized Apollo data sources. Once a data accessor has been refactored, the changes must be deployed so that users can experience improved performance.

3.1 Overview of Data Source Refactor

Inside the Instant Book platform, there is a backend service called csn-venue-core which contains seven different data accessors as shown in Figure 1.

#	Data Access Module
1	Reservation/Booking Service
2	Config
3	Guest Room
4	Passkey
5	Venue Profile
6	Venue Search
7	Amadeus

Figure 1: Data Access Modules in csn-venue-core

In each of these data accessors, there is a variety of different fields that can be accessed via IDs. These fields are accessed in csn-venue-core by different resolvers. To increase the overall performance of the Instant Book platform, each of these data accessors need to be replaced with Apollo data source classes that contain optimized methods corresponding to each field. After refactoring a data accessor, each corresponding resolver needs to be updated to utilize the new data source and the data source needs to be added to the Apollo server. For example, the Config data access module has an IBK Venue Config field that is accessed via Config ID. With my proposed design, the new data source class would have a method that obtains the config field through an optimized GraphQL query. Additionally, the two resolvers which use this module, ibkVenueAggregations and ibkVenueSearch, would be refactored to use this new data source. A prototype of this Config data source class is highlighted in Figure 2.

```
class ConfigAPI extends RESTDataSource {

    // sets base URL
    constructor() {}

    // calls this.get() and passes in configID
    async getIBKVenueConfig(configID) {}

}
```

Figure 2: Class Prototype for Config Data Source

3.2 Overview of Deployment Process

After completing the data source refactor, the next step was to get the changes integrated into the Instant Book platform. First, I submitted the modified csn-venue-core to my team for review in a pull request. After applying their feedback and verifying the service built successfully via integration testing, I merged the refactor into the main code base. Next, I deployed the changes to a silo server and executed manual testing to ensure the platform was still functioning properly. Finally, the refactor was deployed to staging and eventually production in the main Instant Book platform.

3.3 Challenges

During the development of the data source refactor, I encountered one major challenge. The csn-venue-core service utilized an older version of the NX framework and NodeJS that does not support the proposed Apollo data source classes and their caching features. Hence, I was forced to upgrade the entire

service to the latest version of NX that has a compatible version of NodeJS. This was a complex change that involved redeploying the cloud infrastructure utilized by the csn-venue-core service. Furthermore, it involved lots of time and communication with the NX team to get the service ready for the refactor.

4. RESULTS

I was able to replace three of seven data access modules with optimized Apollo data source classes as shown in Figure 3.

Data Access Module	Completed
Reservation/Booking Service	✓
Config	✓
Guest Room	
Passkey	
Venue Profile	
Venue Search	✓
Amadeus	

Figure 3: Overview of Completed Refactors

These new, optimized Apollo data source classes have been deployed to production and are being used by current customers who pay for the Instant Book platform. Since the deployment of these data source classes, overall efficiency of the Instant Book platform has increased by 30%. Users are experiencing faster load times when searching for venues and trying to book them. Furthermore, Cvent has experienced higher rates of user retention since deploying these data source classes to production. These changes will also help Cvent secure new customers who are looking for software to instantly find/book venues.

5. CONCLUSION

During my time at Cvent, I streamlined the data fetching process of the Instant Book platform. Not only did the Apollo data source refactor succeed in making the platform faster overall, but it also modernized the platform and improved user experience. The updated Instant Book platform efficiently obtains data from external sources. More specifically, the optimized Apollo data source classes allow GraphQL to resolve data from REST APIs much faster than before. This allows users to find and book venues quickly. Due to the enhanced performance, more users booked venues through the Instant Book platform. This highlights a major takeaway that I realized after completing this project: optimization is the key to success in full-stack development. Cvent began experiencing greater overall success with the Instant Book platform after the optimization of its existing data accessors. The number of users increased along with revenue and user satisfaction. Hence, it is evident that future optimizations can improve performance and success even more.

6. FUTURE WORK

As shown in Figure 3, only three of seven data access modules were replaced with optimized Apollo data source classes. To complete the refactor, the remaining four data access modules need to be replaced. This will help maximize the overall performance of the Instant Book platform. Once these modules are refactored, the changes will have to be deployed to production so that users can experience improved performance. Furthermore, to ensure that the Instant Book platform is up to date and in accordance with industry best practices, these new Apollo data source classes will have to be maintained. One drawback of my design is that it only focuses on backend optimization. In a full-stack application, it is important to optimize both frontend and backend performance to maximize efficiency. Hence, frontend optimization needs to be added to the design to fully maximize the capabilities of the Instant Book platform.

REFERENCES

[1] Cochrane, B. and Debrunner, D. 2022. GraphQL Optimization: It's More than N+1. (June 2022). Retrieved September 23, 2022 from https://thenewstack.io/graphql-optimization-its-more-than-n1/

[2] Sands-Ramshaw, L. 2020. A Deep Dive on Apollo Data Sources. (April 2020). Retrieved September 23, 2022 from https://blog.graphql.guide/a-deep-dive-on-apollo-data-sources-778618ce06d2