Performance in Apollo Federation — A Controlled Experiment Evaluating the Effects of Execution Strategies and Number of Subgraphs

Anna Bergvall

Bachelor of Science in Engineering, Computer Science and Engineering at LTH, Lund University

Introduction

Traditional RESTful APIs have shown problems with over fetching and under fetching. GraphQL is a technology– and language agnostic specification for retrieving data in a declarative manner, which is a solution to over– and under fetching. GraphQL can be used in two ways: first, to define a schema of a data model, and second, to send queries to this model.

Apollo Federation is an open architecture that allows connecting several different GraphQL APIs to a gateway server, and by doing so creating a unified super graph API where all data defined in the connected subgraphs are available through one single port. For the benefit of Capgemini Malmö, the performance of execution strategies in Apollo Federation was investigated.

Research Objectives

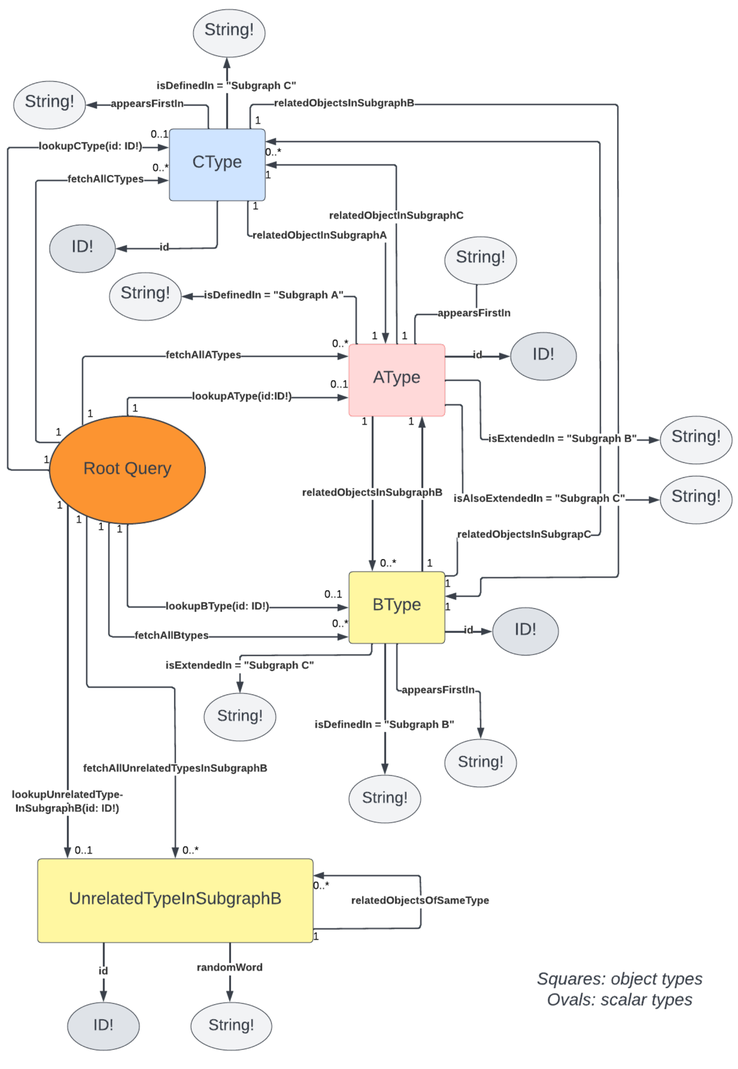
The main goal of the thesis was to answer the following questions:

- How is the performance in Apollo Federation affected by the choice of execution strategy?
- To what extent does the number of subgraphs affect performance in Apollo Federation?

Method

A test application, which data model can be seen in **Figure 1**, was developed in Java, using graphql-java and Spring Boot, in order to answer the research questions above. The microservices were connected through an @ApolloGateway server written in JavaScript to form a unified supergraph adhering to the Apollo Federation specification.

Figure 1:Data model for the federated graph developed in this thesis.



Introduction

Traditional RESTful APIs have shown problems with over fetching and under fetching. GraphQL is a technology– and language agnostic specification for retrieving data in a declarative manner, which is a solution to over– and under fetching. GraphQL can be used in two ways: first, to define a schema of a data model, and second, to send queries to this model.

Apollo Federation is an open architecture that allows connecting several different GraphQL APIs to a gateway server, and by doing so creating a unified super graph API where all data defined in the connected subgraphs are available through one single port. For the benefit of Capgemini Malmö, the performance of execution strategies in Apollo Federation was investigated.

Research Objectives

The main goal of the thesis was to answer the following questions:

- How is the performance in Apollo Federation affected by the choice of execution strategy?
- To what extent does the number of subgraphs affect performance in Apollo Federation?

Method

A test application, which data model can be seen in **Figure 1**, was developed in Java, using graphql-java and Spring Boot, in order to answer the research questions above. The microservices were connected through an @ApolloGateway server written in JavaScript to form a unified supergraph adhering to the Apollo Federation specification.

Three execution strategies were evaluated:

- AsyncExecutionStrategy (”default”)
- AsyncSerialExecutionStrategy (”serial”)
- AsyncExecutionStrategy with parallel data-fetchers (“concurrent”)

To answer the second research question, a set of identical queries were sent to the gateway server connecting one, two, and three federated subgraphs. The testing was conducted in Postman.

Results

The results for the three execution strategies are presented in **Figure 2**. In **Figure 3**, response times are shown for the same set of queries sent to the gateway server with either two or three connected subgraphs.

Figure 2:Results for queries sent to Gateway (A + B)

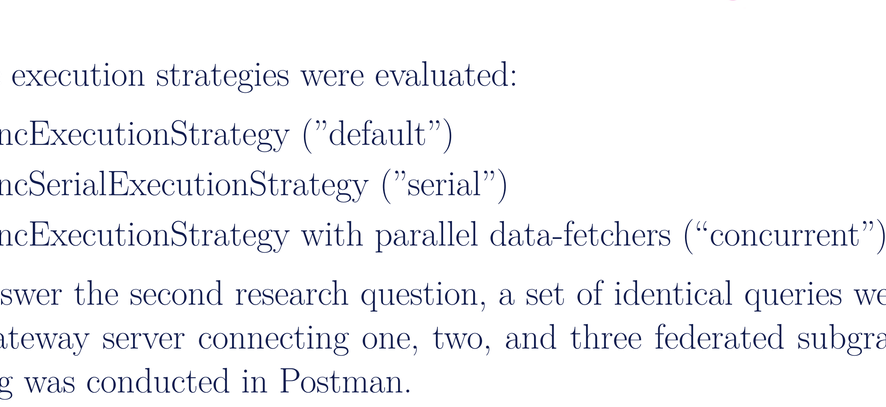
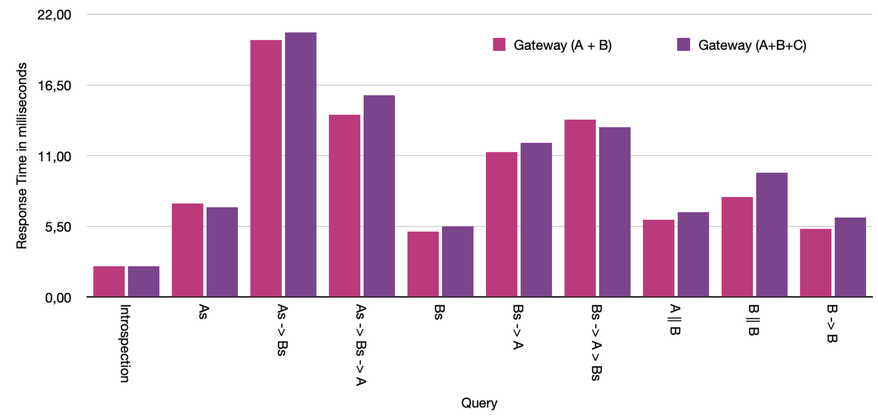


Figure 3:Results for AsyncExecutionStrategy (”standard”) for queries sent to Apollo Federation gateway with two (Gateway (A + B)) respective three (Gateway (A + B + C)) subgraphs



Discussion and Conclusion

The results suggest that AsyncExecutionStrategy offers best overall performance. Moreover, based on these results, the choice of execution strategy matters more with an increased query depth. In addition, the results indicate that concurrent data fetchers are not useful unless queries are rather complex, involving several data fetchers, since starting separate threads comes at a cost. AsyncSerialExecutionStrategy, on the other hand, appears to work well for shallow and narrow queries, but will not offer the best performance for deeper queries based on the these results. However, it is not possible to draw any conclusions about a relationship between the number of subgraphs in a federated architecture and performance with respect to latency as the test results show no clear pattern.