

An Experiment to Measure and Evaluate Performance Improvements on a Spring Boot API

1. Abstract

Application load time is a metric often taken for granted by the modern software user. This investigation details a wealth of literature on the subject of code and application optimizations, before experimentally determining a fitting software solution for a user reported loading time issue on a productionised Spring Boot API. It concisely evaluates a number of potential software improvements and provides advice to engineers about simple methods to improve performance.

2. Introduction

Data Services provides an ordering and administration platform for GCP hosted collections of permissions called IAM Collections. Particularly slow loading time is observed for a page allowing the administration of these IAMs. This project aims to improve this loading time experimentally.

3. Aims and Objectives:

Aim

To demonstrate and evaluate the implementation of various code optimisations for the Data Services My IAM Collection administration page, with a single solution proposed to be deployed.

Objectives

- To investigate the state of current and past literature to guide the development and testing process.
- To design and implement an array of working solutions which completely or partially meet the requirements provided by Data Services.
- To evaluate the effectiveness of each solution to determine the most fitting implementation for deployment.

4. Literature Review:

Research Questions

To understand the most general principles of how and when to optimize a piece of logic or wider system.

How are data models designed to promote efficiency within a system.

How in the specific case of Spring Boot / JPA / Hibernate are data models capable of efficient loading.

What testing methodologies and tools exist to measure and evaluate the speed of a system.

Key Literature

J. Bentley, Writing Efficient Code, Pittsburgh: Carnegie-Mellon University, 1981.

J. Yang and C. Yan, "How not to structure your database-backed web applications:," in 2018 ACM/IEEE 40th International Conference on Software Engineering, Gothenburg , 2018.

A. Leonard, Spring Boot Persistence Best Practices, Romania: Springer Nature, 2020.

Dynatrace, "Modern Cloud Done Right," 2023. [Online]. Available: <https://www.dynatrace.com/>. [Accessed 03 Apr 2023].

Literature Conclusions

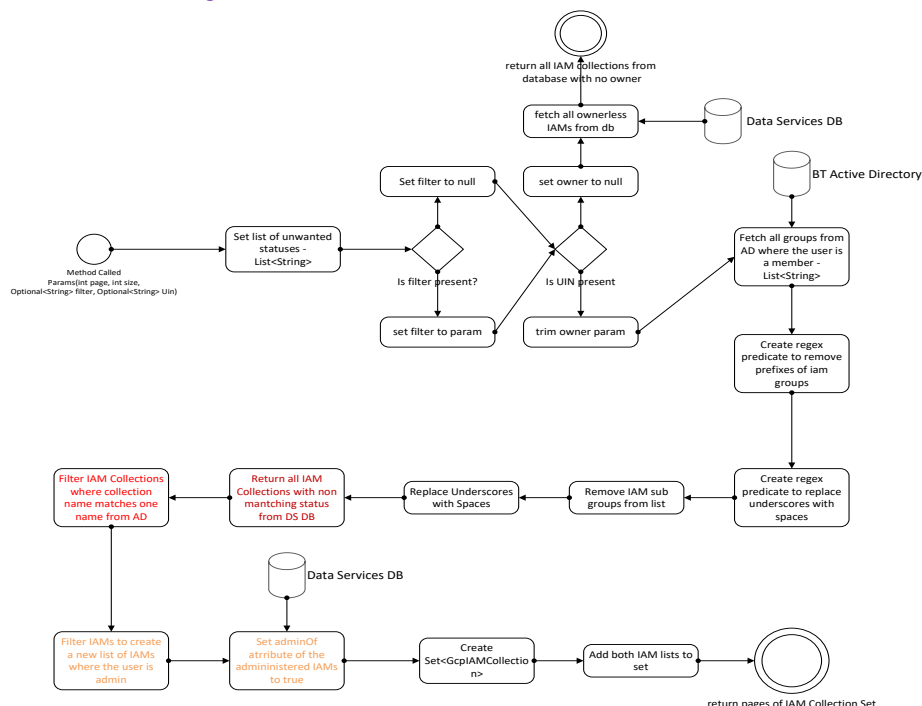
Improvements to the data model can reduce the time taken to load it.

Using lazy loading to avoid over fetching can have large improvements.

Considerations as to how a program will scale are imperative to efficient data loading.

5. Current Implementation:

The current algorithm to serve IAM collection information to the user is shown below:

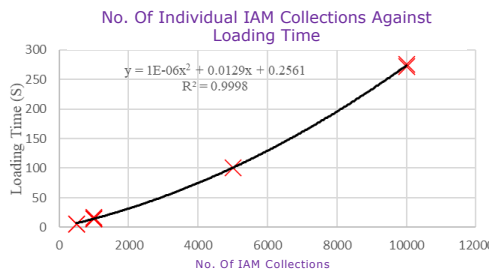


Sections highlighted relate directly to potential solutions, however some experiments require a complete redesign of the system.

6. Current Performance:

The table below shows sample data of the state of the API prior to the project, with an average loading time of around 150 seconds. .

Start Time	Name	Response Time	Service
21/06/2023 11:24:24	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	2 min 37.5 s	GcpIamCollectionController
21/06/2023 09:00:34	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	2 min 34 s	GcpIamCollectionController
21/06/2023 10:09:41	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	2 min 26 s	GcpIamCollectionController
21/06/2023 13:21:59	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	2 min 21.5 s	GcpIamCollectionController
21/06/2023 10:54:44	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	2 min 21 s	GcpIamCollectionController



As part of the project a testing suite was developed to measure the loading time of the current implementation and each proposed solution. The graph to the left shows the results from the project testing suite for the current implementation.

7. Experimental Procedure:

A series of experiments were conducted using a different solution/solution combination. Each experiment conformed to the below requirements:

Requirement 1 –The experiments and chosen solution should attempt to reduce the loading time. Furthermore, users with no requests should experience significantly faster loading times.

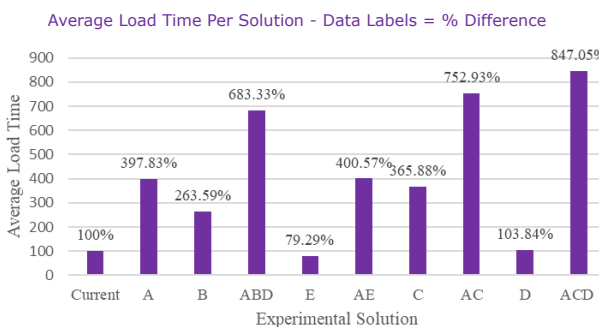
Requirement 2 –The experiments and chosen solution must adhere to the DS coding standards and ultimately maintenance standards. Furthermore, no modifications made should disable previous code from functioning.

Requirement 3 – The chosen solution must not continue to be linked to Active Directory

The following experiments were conducted with their loading times measured across a range of scenarios:

- Lazy Loading Binding Between IAM Collection and Project – **A**
- IAM Collection loading method refactor to Database source of truth– **B**
- Modify IAM Collection Search Query to be more efficient – **C**
- Modification of Admin Group Determination – **D**
- My IAM Collections Data Transfer Object – **E**
- Combined Database driven solution, Lazy Loading and Refined Admin Logic **ABD**
- Combining Lazy Loading with IAM Collection DTO – **AE**
- Combined Lazy Loading with A More Efficient Query – **AC**
- Combining Lazy Loading, More Efficient Selection Query and Improved Admin Logic – **ACD**

8. Results:



After extensive experimentation the table to the left shows the resultant average loading time per experiment. Due to the large number of results for each, which are not feasible to include, the averages provide ample indication as to the performance increase of each solution.

9. Practical Outcome:

The ABD Solution was deployed on 4th July 2023. The table below shows the immediate and highly significant improvements made to page loading times.

Start Time	Name	Response Time	Service
04/07/2023 16:45:51	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	259 ms	GcpIamCollectionController
04/07/2023 16:45:42	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	632 ms	GcpIamCollectionController
04/07/2023 15:41:23	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	2 min 22.1 s	GcpIamCollectionController
04/07/2023 12:40:28	getIAMCollections /api/v1/request/gcp/iam-collection?page=*****&size=*****&uin=*****&filter=	2 min 15.8 s	GcpIamCollectionController

Significant financial gains are made through the deployment of the ABD solution. . Given a 37.5-hour week of 7.5 hours per day with an average cloud engineer cost of £400 per day, a 120 second loading time with an average of 50 queries per day, it is estimated that it would cost £32500 a year to load the page, ignoring any variance in these values.

10. Conclusions:

Combining multiple experimental solutions provided the best overall increase in loading time.

As predicted in the literature, out of the possible solutions, Lazy Loading provided the largest jump in performance per applicable request.

A significant cost saving has been made through the deployment of a solution.

Improvements to the underlying system serving IAM collections yielded a large improvement for Requirement 1, in that users with no IAM collections, experience instant loading time.

Recommendations

Attempts could be made to improve the external validity of the results. The experiments undertaken are extremely specific to Data Services, this could be repeated using a more generic system.

The experiments are specific to the Java layer of the application, improvements to the frontend layer (perhaps pagination), database or infrastructure could be investigated to provide further guidance for developers

Analysis of the cost of improving a feature when compared to the costs due to its' hinderance could provide a unique result set useful to software architects and product owners.