Optimizing IT Infrastructure For A Leading Travel Insurance Company

Executive Summary

As a leading travel insurance company, our client has to manage large volumes of data, users, and services. With frequent crashes of their servers due to multiple factors, the company sought technological solutions to resolve the issue and improve application performance. The HashedIn team used a two-phased approach to identify the root cause of the server crashes and implemented a robust software solution to optimize their existing IT infrastructure.

Problem Statement

When it comes to travel insurance business, success depends on how well you manage the customer relationship. As one of the leading travel insurance companies, our client's mission is to deliver flawless customer service. Managing billions of transactions, with multiple service partners, spread across different geographies is not an easy task.

To handle large volumes of user data and transactions, the application servers should be powerful and reliable. Faced with frequent crashes of their existing servers, the client turned to HashedIn to engineer software solutions to transform their IT infrastructure.

Business Requirements

End Objective

The key requirement was to revamp the existing system of the travel insurance company to make it capable of handling all levels of load throughout the day.

Key Requirements

A solution was developed to enable their team to do the following:

- Prevent servers from crashing and memory leaks.
- Optimize the entire application.
- Eliminate regression in the present functionality

Solution Approach

Our Solution Structure

The first phase of our solution approach involved identifying the possible causes for the shutdown of their system and formulating the recommendations for solving the issues. In phase two, we implemented these proposed solutions to improve the performance of the application.

Phase 1 Solution

First, we explored the different areas of the project to find the problem areas that could possibly cause the system crash. We did a deep analysis of the whole code base and service flow to find the key problem areas:

- HornetQ Migration
- Alfresco (Caching Review)
- Memory leaks + caching
- Security Findings
- Pricing API stability
- Database Index Analysis

Phase 2 Solution

Then we worked on implementing the recommendations to tackle the problem areas. We migrated HornetQ to SQS to deal with the memory/thread leaks. We reduced the in-memory size of pricing spreadsheets and replaced ConcurrentHashMap with EHCache. Their indexes and wildcard searches were modified to optimize queries.

Alfresco was the only single point of failure in case of heavy load. We successfully removed the dependency from alfresco using different caching techniques:

- Saving the file to disk once downloaded so that no unnecessary visits are made to alfresco.
- Using nginx proxy as a gateway to alfresco which can cache the frequently used files.

Technology Stack

- **Back end:** Java EJB, Spring boot, MySQL, AWS with java

Business Outcomes

Our solution empowered the team to prevent system crash and deliver uninterrupted high-quality customer service. Here are the key benefits of our optimization techniques:

- With HornetQ migration to SQS, the team saved a huge amount of memory.
- They could successfully remove the dependency from alfresco.
- The team was able to effectively tackle the memory leaks.
- Their queries were optimized for better results.

HashedIn has helped many promising firms across the globe by building customized solutions to give the users a completely hassle-free experience. Kindly let us know if you have any specific problem/use case, where we can provide more information or consult you.

https://hashedin.com/contact-us/