COMPUTATIONAL PROBLEMS OF ELECTRICAL ENGINEERING

Vol. 14, No. 1, 2024

LAUNCHING MVPS FOR IT STARTUPS: PYTHON/DJANGO/ POSTGRESQL IN LOCATION-TRACKING APPLICATIONS

Volodymyr Franiv, Sviatoslav Zastavskyi, Oleh Kushnir
Department of Optoelectronics and Information Technologies, Ivan Franko National University, Lviv, Ukraine
volodymyr.franiv@lnu.edu.ua, *sviatoslav.zastavskyi@lnu.edu.ua*, *oleh.kushnir@lnu.edu.ua*
https://doi.org/10.23939/jcpee2024/01/001

Abstract: In this study, we develop and evaluate performance of a location-tracking application built using a Python/Django/PostgreSQL stack. Through rigorous experimentation and its analysis, we scrutinize the efficiency of WebSocket connections in facilitating real-time communication and data retrieval. By comparing our findings with the previous research results available on an ASP.NET stack, we contribute to understanding of the technologies for stack selection and the strategies used for optimization of the location-tracking applications. Our results offer valuable guidance for the developers striving to create robust and scalable solutions in this burgeoning domain.

Key words: software development, location tracking, Python, Django, PostgreSQL, ASP.NET, WebSocket, HTTP.

1. Introduction

In today's fast-paced digital landscape, a demand for robust and efficient location-tracking applications (LTAs) continues to soar across diverse fields, e.g. logistics, agriculture and security. These applications play a pivotal role in optimizing processes, enhance security measures and enable informed decision-making by providing real-time insights into the movement of objects [2–5]. With the proliferation of startups in different technology sectors, selecting the right technology stack for developing such applications becomes crucial for ensuring their scalability, flexibility and cost-effectiveness, especially in the context of recent trends of making so-called Minimum Viable Products [5–7].

Since the majority of IT startups begin with a minimum viable product, the key problem is a choice of a proper technological stack for implementing the LTA. It is well known that the problem of operation speed is not viable at the stage of minimum viable products and, as a matter of fact, the actual operation speed would be given by the basic settings of a technological stack chosen by a developer. As a result, we will concentrate on possible alternatives of the technological stacks known from the literature and analyze their viability.

The objective of this study is a performance analysis of the LTA developed using the Python language and the Django framework, with PostgreSQL serving as a database solution. We utilize an Apache JMeter as a performance-testing tool to evaluate the efficiency of this technology stack for the MVP development. Our investigation also aims to provide valuable insights for the IT startups seeking optimal solutions for the LTAs, particularly in comparison to ASP.NET stack elaborated earlier [1]. The LTAs must efficiently receive, store and retrieve the location coordinates from the multiple objects in real time. Therefore, a choice of the technology stack influences significantly the development process, scalability and the overall performance of the LTAs. By using WebSocket connections in this study, we aim to highlight a key distinction of our application from the ASP.NET stack which utilizes HTTP connections [8–12].

Through rigorous performance testing and analysis, we provide a comprehensive insight into suitability of the Python/Django/PostgreSQL stack for LTA development in the context of MVPs. By comparing this stack with the known ASP.NET stack, we identify the strengths and limitations of each approach, thereby enabling startups to make informed decisions regarding the choice of technologies for the MVP development. Finally, by conducting this comparative analysis, we contribute to the body of knowledge related to technology-stack selection and facilitate informed decision-making in the development of LTA solutions.

2. The problem under consideration

The LTA technologies have evolved rapidly in recent years, driven by advances in GPS, IoT and wireless communication technologies. Studies such as those by F. Ahmed et al. [2] have highlighted the importance of seamless asset location and tracking technologies across various domains. These technologies enable real-time monitoring of assets, vehicles and personnel, thereby facilitating efficient object location, route optimization, and risk mitigation.

The choice of database-management system impacts notably the performance and scalability of the LTAs. Relational databases such as MySQL, PostgreSQL and MSSQL have traditionally been popular choices for storing the structured data due to their robustness and support for ACID properties [8–11]. However, non-relational databases such as MongoDB and Redis have gained much prominence for their flexibility, scalability and ability to handle large volumes of the unstructured data [9–11].

Performance testing plays a crucial role in assessing the efficiency and the scalability of the LTAs. Means such as Apache JMeter have become standard tools for conducting performance testing. They enable developers to simulate various usage scenarios and measure key metrics such as throughput, response times, and resource utilization [7]. The studies by S. Matam and J. Jain [8] have demonstrated the efficiency of Apache JMeter in both evaluating web-application performance and identifying performance bottlenecks.

The IT startups face unique challenges when selecting the technology stacks for developing their LTAs. M. Akkaya [3] emphasizes the importance of startup valuation and the role of technology choices in driving business growth and success. The studies by M. Smorgun [7] have highlighted the benefits of using the frameworks like ASP.NET and the tools like Apache JMeter in accelerating application development and ensuring its scalability and reliability.

Earlier, K. Fraczek and M. Plechawska-Wojcik [15] have compared the relational and non-relational databases for web applications. In the studies, the performance, scalability and ease-of-development of different database solutions have been compared. However, there is a strong need for further and deeper comparisons of the technology stacks across different programming languages and frameworks, in particular in the context of the LTAs.

The LTAs often rely on the network protocols to communicate between client and server components. Traditionally, HTTP (Hypertext Transfer Protocol) has been the standard for communication over the web. HTTP is a request–response protocol where the client sends a request to the server and the server responds with the requested data. This protocol is widely used for transmitting data between web servers and web browsers and is well suited for the scenarios where data is exchanged intermittently or in discrete transactions [15–19].

However, with the growing demand for real-time and interactive applications, WebSocket technology has gained a great interest. WebSocket is a communication protocol that provides full-duplex communication channels over a single, long-term connection between a client and a server. Unlike HTTP, which follows a request–response pattern, WebSocket provides bidirectional communication and allows both the client and the server to initiate data exchange at any time [15–21].

WebSocket offers several advantages over the traditional HTTP connections, especially in the scenarios where real-time data updates are critical, such as LTAs. By establishing a persistent connection between the client and the server, WebSocket reduces latency and overhead associated with establishing multiple HTTP connections for each interaction. This persistent connection also enables efficient data streaming, making WebSocket ideal for applications that require continuous data updates, such as live location tracking [3]. In contrast, although the HTTP connections are suitable for transmitting discrete data requests and responses, they may incur additional overhead and latency, especially in the scenarios where frequent data updates are required. As such, the choice between the HTTP and WebSocket connections depends on specific requirements for the applications, including such factors as latency sensitivity, data volume, and the need for bidirectional communication [4].

In the context of LTAs, the choice between HTTP and WebSocket connections can significantly impact their performance, scalability and user experience. Therefore, understanding the differences and implications of these communication protocols is essential for developers when designing and implementing LTA solutions.

By integrating the studies on the network protocols and the communication technologies, we aim to provide a comprehensive understanding of the technological considerations involved in developing the LTAs. In particular, we will perform a comparative analysis of the Python/Django/ PostgreSQL stack with the ASP.NET stack studied previously. This should enable the developers to make informed decisions about the technology selection and the implementation strategies.

3. Experimental results and their analysis

When developing a robust and efficient LTA solution leveraging the Python/Django/ PostgreSQL stack, our team has attempted seamlessly handling the WebSocket connections and efficiently storing the location coordinates in real time. Being inspired by the previous study utilizing the ASP.NET framework [1], we have sought to explore the capabilities of this alternative technology stack in meeting the demanding requirements of the LTAs.

The LTA developed using the Python/Django framework has been meticulously designed to meet the diverse needs of tracking various objects. Using Django’s model-view-template architecture, we have designed a robust backend capable of easily handling the WebSocket connections. Our application integrates seamlessly the WebSocket communication, allowing new location coordinates to be continuously added as they have been received. This approach ensures accurate tracking of each object’s movement, similar to the ASP.NET-based technology.

The core functionality of our application is the efficient handling of WebSocket connections for the real-time communication between the client and the server. These connections facilitate bidirectional communication and enable the server to send updates to the client in real time with no overhead of multiple HTTP requests. Using rich WebSocket handling capabilities of Django Channels, the application processes these connections swiftly and ensures real-time monitoring of object location and seamless access to the record of earlier data.

Our decision to build the LTA on the Python/Django framework was stipulated by several factors that are somewhat similar to those considered by the authors [1] when selecting the ASP.NET framework. The simplicity and readability of Python, as well as the features built in Django for rapid development and scalability made our solution an ideal choice for the startups seeking a cost-effective and scalable technology stack. Additionally, the open-source nature of Python and Django facilitates active community support, thus ensuring continuous improvements and updates.

To check the efficiency of LTA, we used the powerful capabilities of Apache JMeter, which is similar to the methodology employed in the study [7]. Apache JMeter plays a pivotal role in generating a wide array of WebSocket messages to evaluate the performance of the application under varying loads. In our experiments, we utilized Apache JMeter not only to generate different WebSocket messages but also to measure the throughput. The latter represents the number of successfully processed requests per second (RPS) for the two types of requests, retrieving the current location of an object (Fig. 1) and retrieving the track of an object (Fig. 2).

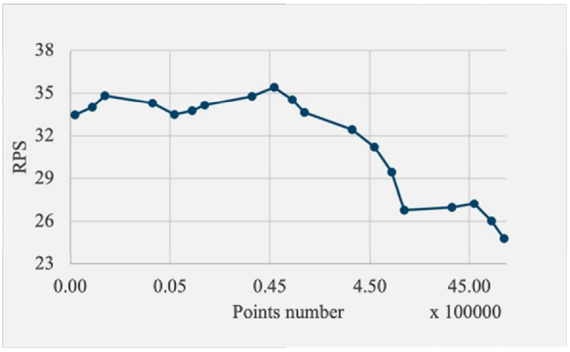


Fig. 1. Dependence of throughput as measured for getting current location of an object on the number of points stored in a database.

Our experimental approach aims at assessing the application's bandwidth under gradual data loading, which is similar to the methodology employed in the study [1]. We increase systematically the amount of data in our database while using the WebSocket messages, thus mimicking real-world usage patterns. A deliberate omission of special indices in the database allows us to gauge the raw performance of our LTA in managing the tracking-data efficiently.

To provide reliable and valid findings, all the experiments were conducted under controlled conditions with standardized hardware and software configurations. This approach minimizes potential variations and confounding factors and enables a robust basis for qualitative analysis of the LTA performance in response to increasing data loads. All the tests were performed on a local machine with 8 GB of RAM, an Intel Core i5 processor, and Ubuntu Linux 18.01 (PostgreSQL 16.2, Python 3.10.12, and Django 5.0).

The WebSocket messages simulate a real-time communication between the client and the server and enable the application to receive and process the requests efficiently. The capabilities of Apache JMeter allow us to simulate various usage scenarios and measure the LTA performance metrics, including the throughput.

In case of a 'get a current object location' request, we measure the LTA ability to retrieve promptly the latest location of a object in real time (see Fig. 1). This metrics deals with the LTA responsiveness and efficiency in handling the requests associated with immediate location updates.

In case of a 'get an object track' request, we evaluate the LTA performance in fetching the complete track of an object, including records of its past locations (Fig. 2). This metrics gauges the LTA ability to efficiently process and retrieve large datasets, thus enabling users to access a complete object-movement dataset.

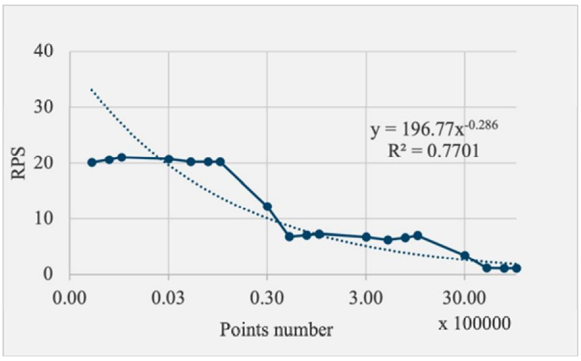


Fig. 2. Dependence of throughput as measured for getting an object track on the number of points stored in a database.

By measuring the throughputs for both types of requests, one can assess the overall performance and scalability of the LTA under varying levels of concurrent users and object-tracking activities. These metrics also enable one to identify potential bottlenecks and optimize the LTA performance.

Following from our observations, we conclude that the throughput plots derived for the Python/Django application based on the WebSocket connections are similar to those obtained earlier for the ASP.NET-based application [1]. In both cases, the experimental curves can be reasonably fitted by a power-law function, which indicates a consistent pattern of the performance scaling with increasing data load.

Our analysis indicates also that, in general, the tracking application based on ASP.NET with the HTTP connections outperforms the Python/Django LTA. In other words, although the Python/Django application utilizes the WebSocket connections, it demonstrates a slightly inferior performance if compared to its ASP.NET-based counterpart.

Our findings suggest that the connection type (either HTTP or WebSocket) might not impact significantly the overall performance of the LTA in case of low load and small amount of data stored in the database. Note in this respect that although the WebSocket connections are often associated with lower latency and overhead, when compared to the traditional HTTP connections, the performance difference between these two types of connections is not substantial in our experiments.

These observations highlight a complex interplay of different factors in determining the overall performance of the LTAs, including the programming languages, the frameworks, the database systems, and the connection types. Further analysis and experimentation would be necessary to understand better the underlying factors driving these performance differences and optimize the LTA accordingly.

Summing up, the LTA developed using the Python/Django/PostgreSQL stack demonstrates performance comparable to its ASP.NET counterpart. It offers efficient handling of WebSocket connections, real-time tracking capabilities and seamless access to the entire dataset.f

4. Conclusions

Our comparative analysis of the LTAs developed using the Python/Django/PostgreSQL stack and the earlier ASP.NET stack [1] sheds light on the dynamics of stack-technology selection and its impact on overall LTA performance. Through rigorous experimentation and analysis, a number of important conclusions have been drawn.

First, our study highlights the critical role of stack technology in determining the scalability, efficiency and responsiveness of the LTAs. Although both the Python/Django/PostgreSQL and ASP.NET stacks offer robust frameworks and database solutions, subtle differences in their performance can still be observed.

Second, our experiments reveal that the choice between HTTP and WebSocket protocols, however significant, could not always yield substantial performance disparities, in particular under low-load and small-dataset conditions. Despite the WebSocket's advantages in reducing latency and overhead, the performance differences between the two protocols observed in our experiments are negligible.

Our findings highlight the importance of a holistic approach to performance optimization that takes into account multiple factors such as programming languages, frameworks, database systems and connection types. The interaction of these ingredients influences the overall performance and scalability of the LTAs.

Our study provides a valuable insight into the choice of stack technology for LTAs and, moreover, it also emphasizes a clear need for further experimentation in the field. Future research should explore such extra factors affecting the LTA performance as optimization techniques, network configurations and hardware resources.

As a result, our comparative analysis contributes to the knowledge referred to the choice of stack technology when developing LTA solutions. By providing valuable empirical evidence, our study should help developers in making informed decisions and optimizing the performance of their LTAs in any real-world scenarios.

References

[1] V. Franiv, S. Vasyluk, O. Biletskyi, and I. Franiv, "An Investigation into the Efficiency of Specific Databases for Tracking Purposes in Scope of IT Startup", in *Proc. IEEE 13th Intl. Conf. on Electronics and Inf. Technol. (ELIT)*, pp. 186–190, Lviv, Ukraine 2023. DOI: 10.1109/ELIT61488.2023.10310719

[2] F. Ahmed, M. Phillips, S. Phillips, and K.-Y. Kim, "Comparative Study of Seamless Asset Location and Tracking Technologies", *Procedia Manufacturing*, vol. 51, pp. 1138–1145, 2020. DOI: 10.1016/j.promfg.2020.10.160

[3] M. Akkaya, *Startup Valuation*, IGI Global, pp. 137–156, 2019.

[4] A. Mathur and H. Agarwal, "Study of Challenges Faced by Startup Industries", *A referred & peer-reviewed quarterly research journal*, vol. 48, pp. 58–67, 2023.

[5] D. Esposito, "Building Web Solutions with ASP.NET and ADO.NET", *Redmond: Microsoft Press*, 2002.

[6] K. Padaliya, "C# Programming with .Net Framework", 2019.

[7] W. S. Vincent, *Django for APIs: Build web APIs with Python and Djang.* New York, USA: WelcomeToCode, 2020.

[8] S. Matam and J. Jain, *Pro Apache JMeter: web application performance testing*, Apress, USA, 2017.

[9] C. H. Lee and Y. L. Zheng, "SQL-to-NoSQL Schema Denormalization and Migration: A Study on Content Management Systems", in *Proc. IEEE Intl. Conf. on Systems*, Man, and Cybernetics (SMC), pp. 2022–2026, 2015. DOI:10.1109/SMC.2015.353

[10] M. Abu Kausar, M. Nasar, and A. Soosaimanickam, "A Study of Performance and Comparison of NoSQL Databases: MongoDB, Cassandra, and Redis Using YCSB", *Indian J. Sci. Technol.*, vol. 15, pp. 1532–1540, 2022. DOI: 10.17485/IJST/v15i31.1352

[11] J. R. Lourenco, B. Cabral, P. Carreiro, M. Vieira, and J. Bernardino, "Choosing the right NoSQL database for the job: a quality attribute evaluation", *J. Big Data*, vol. 2, pp. 1–26, 2015. DOI: 10.1186/s40537-015-0025-0

[12] L. Vokorokos, M. Uchnar, and L. Lescisin, "Performance optimization of applications based on non-relational databases", in *Proc. Conf. on Emerging eLearning Technol. and Appl. (ICETA)*, pp. 371–376, 2016. DOI: 10.1109/ICETA.2016.7802079

[13] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain, “A survey and comparison of relational and non-relational database”, *Intl. J. Engin. Res. & Technol*., vol. 1, pp. 104–118, 2012.

[14] J. Han, E. Haihong, G. Le, and J. Du, “Survey on NoSQL database”, in *Proc. 6th Intl. Conf. on Pervasive Comput. and Appl. (ICPCA)*, pp. 363–366, 2011.

[15] K. Fraczek and M. Plechawska-Wojcik, “Comparative analysis of relational and non-relational databases in the context of performance in web applications”, *in Proc. Conf.: Beyond Databases, Architectures and Structures*, pp. 205–213, 2017.

[16] S. Gupta and G. Narsimha, “Efficient Query Analysis and Performance Evaluation of the Nosql Data Store for Big Data”, in *Proc. 1st Intl. Conf. on Comput. Intelligence and Informatics. Springer (Singapore)*, pp. 549–558, 2017.

[17] K. Chodorow and M. Dirolf, *MongoDB: The Definitive Guide*, O’Reilly Media, 2010.

[18] D. Sullivan, *NoSQL for Mere Mortals*, Addison-Wesley, 2015.

[19] E. Brewer, “CAP twelve years later: how the rules have changed”, *Computer*, vol. 45, pp. 23–29, 2012.

[20] X. Li, Z. Ma, and H. Chen, “QODM: A query-oriented data modeling approach for NoSQL databases”, *Advanced Research and Technology in Industry Applications (WARTIA)*, pp. 338–345, 2014.

[21] Y. Li and S. Manoharan, “A performance comparison of SQL and NoSQL databases”, in *Proc. IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing (PACRIM)*, pp. 15–19, 2013.

МІНІМАЛЬНО ЖИТТЄЗДАТНІ ПРОДУКТИ ДЛЯ ІТ-СТАРТАПІВ: ВИКОРИСТАННЯ PYTHON/DJANGO/ POSTGRESQL У ПРОГРАМНИХ ДОДАТКАХ ВІДСТЕЖЕННЯ МІСЦЕЗНАХОДЖЕННЯ

Володимир Франів, Святослав Заставський,
Олег Кушнір

Додатки для відстеження місцезнаходження (ДВМ) відіграють важливу роль у таких сферах, як логістика, сільське господарство та безпека, забезпечуючи реальні дані про рух ресурсів та активів. Зі зростанням кількості стартапів у секторі технологій вибір відповідного технологічного стека стає вирішальним для забезпечення масштабованості, гнучкості та вартості додатків, особливо в актуальному нині контексті “мінімально життєздатних продуктів”.

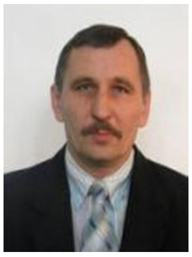
У цьому дослідженні здійснено розроблення та оцінено продуктивність ДВМ, який використовує стек технологій Python/Django/PostgreSQL. Виконавши ретельні експерименти та їхній аналіз, ми порівняли ефективність цього технологічного стека зі стеком ASP.NET, досліджено у попередніх працях. Наше дослідження спрямоване на одержання інформації щодо вибору технологічних стеків і стратегій оптимізації для програмних ДВМ.

У результаті вимірювань одержано інформацію стосовно швидкодії ДВМ, які грунтуються. на Python/Django/ PostgreSQL. Акцентуючи увагу на використанні WebSocket-з’єднань, ми висвітлили переваги та недоліки цього підходу, порівняно з традиційними HTTP-з’єднаннями. Дослідження передбачало тестування продуктивності ДВМ за допомогою Apache JMeter для розуміння придатності стека технологій Python/Django/PostgreSQL для розробки ДВМ.

Загалом наше дослідження сприятиме розвитку галузі вибору технологічних стеків для стартапів і допоможе в прийнятті обгрунтованих рішень розробників ДВМ. Подавши низку емпіричних результатів і їхнє узагальнення, ми прагнули надати можливість розробникам приймати обгрунтовані рішення та оптимізувати продуктивність своїх ДВМ у реальних умовах.

Volodymyr Franiv. PhD, Associate Professor at Optoelectronics and Information Technologies Department, Faculty of Electronics and Computer Science, Ivan Franko National University of Lviv. Research interests: Computer Science, Artificial Intelligence. Senior Software Engeneer with more than 10 years experience in Software Development.

Sviatoslav Zastavskyi. Student at Ivan Franko National University of Lviv. Research interests: Computer Science.



Oleh Kushnir. Professor, Doctor of Sciences, Head of Optoelectronics and Information Technologies Department, Faculty of Electronics and Computer Science, Ivan Franko Nationanl University of Lviv. Research interests: Computational Linguistics, Natural Language Processing, Complex Systems and Networks, Information Retrieval.

Received: 02.04.2024, Accepted: 25.04.2024
ORCID ID: 0000-0001-9856-1962 (V. Franiv)
ORCID ID: (S. Zastavskyi)
ORCID ID: 0000-0002-1545-7666 (O. Kushnir)