A Cloud-Based Solution for Multi-Agent Traffic Control Systems

Extended Abstract

Chikadibia Ihejimba
The University of Texas at Dallas
Richardson, United States
cki100020@utdallas.edu

Behnam Torabi
The University of Texas at Dallas
Richardson, United States
behnam.torabi@utdallas.edu

Rym Z. Wenkstern
The University of Texas at Dallas
Richardson, United States
rymw@utdallas.edu

ABSTRACT

This paper introduces a cloud-based solution for multi-agent Traffic Control Systems (TCSs). We focus on re-architecting the *DALI* system, a self-adaptive, collaborative multi-agent TCS. We explore different options to effectively engineer and deploy a highly available, low-latency cloud-based solution for *DALI*.

KEYWORDS

Traffic Control Systems; Cloud-Native; Microservices; Serverless

ACM Reference Format:
Chikadibia Ihejimba, Behnam Torabi, and Rym Z. Wenkstern. 2023. A Cloud-Based Solution for Multi-Agent Traffic Control Systems : Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.

1 INTRODUCTION

Over the years, a plethora of solutions for Traffic Control Systems (TCSs) has been proposed. Most research in TCS focuses on the intricacies of the traffic system algorithms without deep consideration of deployment options. As such, most traffic lights are still managed by intersection controllers connected directly or indirectly to a higher-level central traffic management unit [7, 11, 18, 19]. Recently, a few agent-based TCSs have been successfully deployed in the US [12, 17]. While the MAS paradigm brings in the much-needed concepts of distribution, intelligence, autonomy, and collaboration, agent-based TCSs are still deployed using conventional approaches: agents are either physically integrated into intersection controllers [12] or run on a data center and connect to intersection controllers via VPN [17]. No existing agent-based TCS in the field takes advantage of modern cloud-native technologies and tools.

This paper presents a cloud-native deployment solution for *DALI* [14–17], a multi-agent, collaborative traffic control system currently operating in the US.

2 RELATED WORKS

Existing TCSs can be classified as non-MAS or MAS, non-cloud-based or cloud-based. Furthermore, cloud-based solutions can be categorized as non-cloud-native or cloud-native. In this section, we restrict our discussion to commercial systems or deployed research systems.

2.1 Non-MAS Solutions

Non-MAS TCSs are conventional systems used to manage traffic. They are either non-cloud based or cloud-based.
Non-MAS, Non-Cloud Based solutions include conventional TCSs deployed in an on-premise data center or on a traditional server. Systems that fit in this category include TRANSYT [11] , SCOOT ([19], and STREAM [18]. It is well known that centralized approaches limit the scalability of TCSs.
Non-MAS, Cloud-Based solutions include the SCATS commercial system [7]. Though the core platform is cloud-based, SCATS still suffers from the limitations of conventional centralized systems.

2.2 MAS Solutions

There are currently limited commercial MAS solutions and very few deployed MAS research solutions for traffic control. Unlike non-MAS solutions, in multi-agent TCSs the intersection controllers are augmented with software agents which are responsible to define and optimize traffic signal timing plans in real-time for their respective intersections.

MAS, Non-Cloud Based multi-agent TCSs leverage the decentralized nature of MAS but run on a traditional data center or in a non-cloud environment. SURTRAC [12] and *DALI* fall into this category. *DALI* [16] is a collaborative multi-agent TCS successfully deployed in 2019 in the US. In *DALI*, agents run on a data center and connect to intersection controllers via VPN. Agent-based solutions provide scalability for TCS, but their non-cloud implementation requires expensive high-speed direct connections for low latency.

MAS, Cloud-Based. Multi-agent TCS in this category take the extra step of utilizing cloud-based technologies.
MAS, Cloud Native System. Cloud-native solutions use modern software development techniques such as microservices, containers, agile methodologies, and DevOps [6] (i.e., combination of practices to unite development and operations) to build resilient and scalable systems. There are currently no MAS, cloud-native systems for traffic control.

In this paper, we propose a novel cloud-native implementation of a multi-agent TCS using serverless computing, containers, serverless database with auto-scalability, and software-defined networking. The solution provides low latency, scalability, and a click-to-deploy DevOps solution for easy deployment to cities and municipalities. The contributions are a native cloud solution, a highly scalable, and low latency solution, a click-to-deploy DevOps solution, and a blueprint for taking non-cloud-based traffic control systems to cloud-native implementation.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

3 SYSTEM DESIGN AND IMPLEMENTATION

3.1 Latency and Scalability

Two primary factors have influenced the design decisions for the proposed system. These are *latency*[9] and *scalability*[13].

Our proposed solution employs AWS microservices tools and AWS Fargate [4] serverless container solution to achieve latency and on scalability, it uses a *microservices architecture*, which allows "a large application to be separated into smaller independent parts, with each part having its own realm of responsibility" [5].

3.2 Cloud Architecture Design

Our approach to developing a novel solution for a cloud-based TCS involved multiple iterative processes and experiments to optimize the solution. Since we were modernizing an existing solution, *DALI*, which runs in an on-premises data center, we followed the recommended industry best practices, cloud-native and the Seven R's framework (Re-hosting, Re-platforming, Re-factor, Replacing, Retiring, Retaining, and Reimagining), for migrating to the cloud [1, 10].

3.3 System Design

For system design, we considered three cloud implementation options for *DALI*:

(1) *Lift & Shift :* focuses on deploying our instances to emulate the current state of *DALI* with no changes - *Re-Host.*
(2) *Container Based Architecture:* combines cloud-native serverless, microservices, and VMs architecture to improve scalability and lower latency - *Re-Host & Re-factor*
(3) *Serverless Container Based Architecture:* is similar to the second option, but the middle tier uses a serverless container - *Re-Platform, Re-factor, & Re-Host.* This offers improved scaling on demand and lower latency without extensive code-rewrite for the middle layer, making it a cost-effective and desirable option for achieving serverless scalability and low latency.

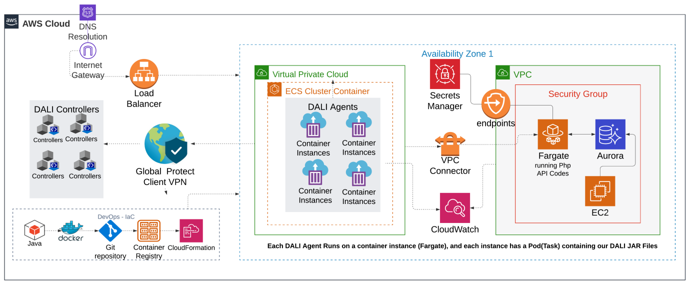

Figure 1: *DALI* Cloud Solution Architecture

4 EXPERIMENTAL RESULTS

4.1 Experiment Setting

Two experiments were conducted to evaluate the effectiveness of the proposed cloud-native solution compared to a traditional EC2 instance using APIMetrics[2] observability tool. Design options 1 and 3 were selected, and seven-day response time was measured along with the system's availability rate to determine if the proposed solution has better latency and scalability.

Experiment 1: EC2. Our REST API and database were hosted on EC2 Instances with the following specifications (t2.medium, 4GB memory, 2 vCPUs, EBS only, 64-bit platform). Our experiment involved running DALI API on an Apache web server, which connects to the controller. A status API call retrieves a data payload from the back-end database.

Experiment 2: AWS Fargate. Our Serverless container solution is deployed on Docker using AWS CloudFormation written in YAML[8]. The PHP code is packaged in the Docker container and deployed to AWS Fargate tasks with specifications similar to EC2. The system is auto-scalable, with new tasks created to handle extra demand. The database is set up on AWS Aurora Serverless[3] with a similar configuration as the EC2 database in experiment 1.

4.2 Results

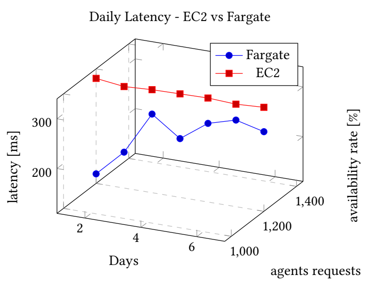

Figure 2: Latency(ms)

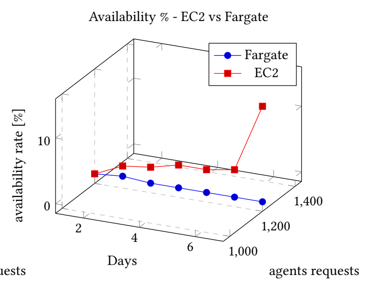

Figure 3: Availability(%)

Latency Results: Experiment 2 performs better than experiment 1, with an average response time, lowest response time, and highest response time improving by 20.83%, 21.15%, and 47.16%, respectively. The average daily latency for EC2 was 317.7ms, whereas Fargate had a latency of 232.7ms, signifying a 36.5% improvement during the seven days of the experiment.
Availability Results: Experiment 2 had a higher availability rate of 99.95% compared to 95.72% in experiment 1. Experiment 1's system logs showed that the system experienced scalability issues due to limited resources to handle requests and "out of memory" issues, resulting in a failure rate of 4.28%.

5 CONCLUSION

This paper presents a cloud-based implementation of DALI, a traffic control system, which solves latency issues and provides on-demand scalability. The solution offers a "click-to-deploy" DevOps feature for easy implementation in cities and municipalities and a blueprint for MAS Cloud implementation. Future work includes extending the implementation to other public cloud providers and a multi-cloud-hybrid deployment.

REFERENCES

[1] Accenture. 2021. Cloud Application & Infrastructure Modernization | Accenture. http://tiny.cc/accenture7rs. (Accessed on 10/16/2022).

[2] Apimetrics. 2022. APImetrics: APImetrics. https://client.apimetrics.io/. (Accessed on 10/27/2022).

[3] AWS. 2023. Amazon Aurora | AWS. https://aws.amazon.com/rds/aurora/. (Accessed on 02/27/2023).

[4] AWS. 2023. Serverless Compute Engine–AWS Fargate–Amazon Web Services. https://aws.amazon.com/fargate/. (Accessed on 02/21/2023).

[5] Google. 2022. What Is Microservices Architecture?| Google Cloud. http://tiny.cc/micro-services. (Accessed on 10/19/2022).

[6] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. 2019. A survey of DevOps concepts and challenges. *ACM Computing Surveys (CSUR)* 52, 6 (2019), 1–35.

[7] NSW. 2022. SCATS. https://www.scats.nsw.gov.au. (Accessed on 10/19/2022).

[8] YAML Organization. 2022. The Official YAML Web Site. https://yaml.org/. (Accessed on 04/14/2022).

[9] Perfmatrix. 2021. *Latency, Bandwidth, Throughput and Response Time.* http://tiny.cc/Perfmatrix (Accessed on 03/30/2022).

[10] Ronen Schwartz. 2021. The 7 R's: Why infrastructure is critical to your application cloud migration. http://tiny.cc/sevenRs. (Accessed on 03/30/2022).

[11] TRL Software. 2022. TRANSYT - TRL Software. https://tinyurl.com/ttransyt. (Accessed on 10/12/2022).

[12] Rapid Flowt Tech. 2022. Surtrac: Intelligent Traffic Signal Control System. https://www.rapidflowtech.com/surtrac. (Accessed on 10/27/2022).

[13] techtarget. 2021. What Is Scalability? - Definition from SearchDataCenter.com. https://www.techtarget.com/searchdatacenter/definition/scalability. (Accessed on 02/27/2023).

[14] Behnam Torabi, Rym Z Wenkstern, and Robert Saylor. 2018. A self-adaptive collaborative multi-agent based traffic signal timing system. In *2018 IEEE International Smart Cities Conference (ISC2)*. IEEE, IEEE, 1–8.

[15] Behnam Torabi, Rym Z Wenkstern, and Robert Saylor. 2020. A collaborative agent-based traffic signal system for highly dynamic traffic conditions. *Autonomous Agents and Multi-Agent Systems* 34, 1 (2020), 1–24.

[16] Behnam Torabi and Rym Zalila-Wenkstern. 2020. DALI: An Agent-Plug-In System to" Smartify" Conventional Traffic Control Systems. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS, 2120–2122.

[17] Behnam Torabi, Rym Zalila-Wenkstern, Robert Saylor, and Patrick Ryan. 2020. Deployment of a Plug-In Multi-Agent System for Traffic Signal Timing. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 1386–1394.

[18] Transmax. 2022. Traffic Services – Transmax. https://tinyurl.com/Transmaxt. (Accessed on 10/13/2022).

[19] trlsoftware. 2022. SCOOT® - TRL Software. https://tinyurl.com/tscoot. (Accessed on 10/12/2022).