Improving Performance Estimation of Smart City Simulations Using the Actor Model

Francisco Wallison Rocha[1], **Emilio Francesquini**[2], **Daniel Cordeiro**[1]

[1]Escola de Artes, Ciências e Humanidades — Universidade de São Paulo (USP)
São Paulo — SP — Brasil

{wallison.rocha, daniel.cordeiro}@usp.br

[2]Centro de Matemática, Computação e Cognição – Universidade Federal do ABC (UFABC)
Santo André — SP — Brasil

e.francesquini@ufabc.edu.br

Abstract. *The United Nations estimates that the world will reach around 10.4 billion people by 2050. Urban mobility problems already faced by large cities will be worsened, such as the emission of polluting gases into the atmosphere. These problems require innovative solutions. Solutions within the context of smart cities emerge as an alternative, an example of which is simulations. However, large-scale simulations are still a challenge. Techniques such as SimEDaPE emerge to help face these challenges. For this reason, they must be robust techniques to deal with a large volume of data. Therefore, this work presents a new approach using the actor-based model to improve the performance of SimEDaPE. The approach proposed here proved to be* $48\times$ *than its predecessors.*

1. Introduction

The world's population has reached 8 billion people on the Earth's globe. The United Nations estimates that the world's population is expected to increase by nearly 2 billion by 2050, peaking at nearly 10.4 billion in the mid-2080s[1]. It is also estimated that by the year 2100, 90% of the global population will live in urban areas. This growth will result in the exacerbation of several problems related to urban mobility, such as excessive carbon dioxide emissions into the atmosphere. Cities are responsible for more than 70% of CO2 emissions [Hong et al. 2022].

Solving such problems is not trivial. They require innovative solutions to solve or mitigate them. Solving such problems is not trivial. They require innovative solutions to resolve or mitigate them. An alternative to mitigate these problems is the use of technologies in the context of smart cities. An example of these technologies are simulations. An example of these simulators is the InterSCSimulator. The InterSCSimulator is a large-scale simulator that simulates a city or regional map with pedestrian and car trips, as well as metro and bus systems [Santana et al. 2017]. Although the InterSCSimulator is a large-scale simulator, it requires improvements and techniques to enhance its performance. An example is the Simulation Estimation by Data Patterns Exploration (SimEDaPE) [Rocha et al. 2021]. SimEDaPE is a technique that uses unsupervised learning to identify pattern recurrence, obtained from previous simulations, to estimate new

[1]https://www.un.org/en/global-issues/population

similar simulations without running them completely. This way, SimEDaPE accelerates InterSCSimulator simulations.

However, large-scale simulations generate significant volumes of data, posing a challenge for SimEDaPE. To expedite simulations, the technique must not become a bottleneck. Efforts have been made to enhance SimEDaPE's performance using multi-threading, as noted by [Rocha et al. 2022a] and [Rocha et al. 2022b]. Nonetheless, the multi-threading model presents various challenges, including race conditions and deadlocks. An alternative to mitigate these challenges is the utilization of the actor model. This model offers isolation, concurrency, and fault tolerance, among other features. Consequently, this work proposes a new implementation alternative to the ones cited in [Rocha et al. 2022a] and [Rocha et al. 2022b], employing the Akka Framework[2]. Akka is an actor model framework implemented in the Scala language, with versions available in Java.

2. Background

Smart city technologies are very useful in solving several urban mobility problems. An example of these technologies is simulations. Simulations allow testing and validating solutions before applying them. An example of a simulator is the InterSCSimulator. It simulates a map of a city or region, including pedestrian and car trips and bus and metro systems. Facing some limitations presented by the InterSCSimulator, such as excessive memory usage and poor load balancing in distributed memory, the big challenge of the InterSCSimulator is to simulate scenarios such as São Paulo with millions of elements. To mitigate or solve these limitations, the Simulation Estimation by Data Patterns Exploration (SimEDaPE) was proposed [Rocha et al. 2021].

The SimEDaPE uses pattern recurrence obtained from previous simulations to estimate new similar simulations without running them completely. SimEDaPE has several steps to carry out the estimate. One of them is the *Warping Path* calculation. As seen in the study by [Rocha et al. 2022b], it is one of the steps that take the most time to execute, corresponding to around 95.21% of SimEDaPE's execution time. This step consists of applying Dynamic Time Warping (DTW) [Berndt and Clifford 1994] to extract the Warping Path (WP). WP is the end-to-end temporal mapping between two time series, representing their similarity and displacement in time.

To improve the performance of this stage, two approaches were proposed in [Rocha et al. 2022a] and [Rocha et al. 2022b]. Both approaches used Python implementations that were faster than the original implementation. These implementations were provided by the DTAIDistance[3] library using Cython in its core. Furthermore, both provided parallel approaches using the Joblib[4] library. The big difference between the two works is that in [Rocha et al. 2022b], load balancing was implemented between the processes. However, the balancing in this case brought few gains in relation to [Rocha et al. 2022a]. Both approaches achieved a $3x$ speedup in relation to the faster sequential implementation, even though in the experiments carried out, an 8-core processor running 8 processes in parallel was used. The cause of this limitation in speedup can

[2]`https://akka.io/`
[3]`https://pypi.org/project/dtaidistance/`
[4]`https://joblib.readthedocs.io/en/stable/`

be explained because Python's Global Interpreter Lock (GIL) restricts the simultaneous progress of threads, hindering the exploitation of multi-core CPU benefits for parallel execution [Mattson et al. 2021].

3. Proposal and Experimental Results

Given the limitations presented by previous approaches, this work proposes a new approach to improve the performance of the Warping Path calculation stage. Alternatively, this work provides an implementation of the step using the Java language, a compiled language that runs on the Java Virtual Machine (JVM). Java offers robust support for multi-threading, especially in the version used in this project, Java 21. Java 21 brings the benefits of virtual threads[5]. Furthermore, for parallelism, this approach implements the actor-based model using the Akka framework. In this model, each actor represents a thread being executed, where communication between authors is done through the exchange of non-blocking messages. Akka, with the actor-based model, offers isolation, concurrency, fault tolerance, and high performance, among other benefits. Another important detail of this approach is the use of an optimized Dynamic Time Warping implementation[6]. This implementation was based on the work by [Salvador and Chan 2007].

In this proposed method, a single management actor is created after the clustering step of SimEDaPE (using a clustering algorithm [Rocha et al. 2021]). This actor is responsible for starting and ending the execution when all Warping Paths (WP) have been calculated. To start processing, this managing actor creates actors for each cluster. Cluster actors hold information related to the cluster, such as time series and centroids (grouped and generated using a clustering algorithm), and create actors responsible for calculating the WP. When these actors finish calculating the WP, they inform their respective cluster actor that they have finished the calculation. Processing ends when all cluster actors inform the managing actor that all WP of their time series have been calculated.

To determine the gains from the new approach, two experiments were carried out. The first experiment was performed to compare the two approaches with 445,501 time series of size 3,403 data points divided into 64 clusters. The time for the new approach was 76.8s and the time for the proposal by [Rocha et al. 2022b] was 3731.65s. The second experiment used a dataset with 6,000,000 time series of size 6,806 data points distributed in 64 clusters. This time series corresponds to around 500,000 street simulations over 24 hours, with time divided into 2 hours. Just completed for the approach of this work, due to memory limitations of the proposal by [Rocha et al. 2022b]. The machine used to run the experiments has a 2-thread AMD EPYC 7453 processor (28 cores/56 threads) with 1 TB of RAM.

4. Conclusions

This work presented a new approach using the actor-based model and implemented in a compiled language with more robust support for multi-threading, as an alternative to the approaches presented in [Rocha et al. 2022a] and [Rocha et al. 2022b]. The experimental results show the great improvements in execution time presented by the new approach proposed here compared to the previous ones. The approach proposed here proved to be

[5]`https://docs.oracle.com/en/java/javase/21/core/virtual-threads.html`
[6]`https://code.google.com/archive/p/fastdtw/`

$48\times$ faster than the previous one. Thus, it is possible to notice significant gains. This increases the robustness of SimEDaPE for estimating large simulations.

By noting the performance presented by out approach, as future work, we see the possibility of implementing it for other stages, such as clustering. An example of using Akka in clustering is presented in the work by [Taamneh et al. 2020]. Furthermore, the proposed approach was only applied to one architecture. So, it would be interesting to apply it to other architectures, such as distributed memory since the framework offers support for this. Another important thing for future work is to stress the experiments more to find out how the approach deals with large volumes of data, thus being able to identify problems with memory management.

Acknowledgments:
This work is part of the project "Trends in High-end Computing Performance, from Resource Management to New Architectures Computers", grant #2019/26702-8, São Paulo Research Foundation (FAPESP).

Referências

Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370, Seattle, WA. AAAI Press.

Hong, S., man Hui, E. C., and Lin, Y. (2022). Relationship between urban spatial structure and carbon emissions: A literature review. *Ecological Indicators*, 144:109456.

Mattson, T. G., Anderson, T. A., and Georgakoudis, G. (2021). Pyomp: Multithreaded parallel programming in python. *Computing in Science Engineering*, 23(6):77–80.

Rocha, F., Francesquini, E., and Cordeiro, D. (2022a). Fast simedape: Simulation estimation by data patterns exploration. In *Anais da XIII Escola Regional de Alto Desempenho de São Paulo*, pages 37–40, Porto Alegre, RS, Brasil. SBC.

Rocha, F., Francesquini, E., and Cordeiro, D. (2022b). Improving smart city simulation performance with simedape and parallelism. In *Anais do XXI Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, pages 108–113, Porto Alegre, RS, Brasil. SBC.

Rocha, F. W., Fukuda, J. C., Francesquini, E., and Cordeiro, D. (2021). Accelerating smart city simulations. *Latin America High Performance Computing Conference*. To publish.

Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.

Santana, E. F. Z., Lago, N., Kon, F., and Milojicic, D. S. (2017). InterSCSimulator: Large-scale traffic simulation in smart cities using erlang. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 211–227. Springer.

Taamneh, S., Qawasmeh, A., and Aljammal, A. H. (2020). Parallel and fault-tolerant k-means clustering based on the actor model. *Multiagent and Grid Systems*, 16(4):379–396.