Scalable Alternative Route Computation with ACE: A C++17 Library for HPC Traffic Simulations

Paulo Silva
paulo.silva@vsb.cz
IT4Innovations, VSB - Technical University of Ostrava
Ostrava, Czech Republic

Pavlína Smolková
pavlina.smolkova@vsb.cz
IT4Innovations, VSB - Technical University of Ostrava
Ostrava, Czech Republic

Kateřina Slaninová
katerina.slaninova@vsb.cz
IT4Innovations, VSB - Technical University of Ostrava
Ostrava, Czech Republic

Jan Martinovič
jan.martinovic@vsb.cz
IT4Innovations, VSB - Technical University of Ostrava
Ostrava, Czech Republic

Matej Špeťko
matej.spetko@vsb.cz
IT4Innovations, VSB - Technical University of Ostrava
Ostrava, Czech Republic

João Barbosa
joao.barbosa@vsb.cz
IT4Innovations, VSB - Technical University of Ostrava
Ostrava, Czech Republic

Emanuele Vitali
CSC - IT Center for Science
Espoo, Finland
emanuele.vitali@csc.fi

Abstract

We present ACE (Asynchronous Communication and Execution), a C++17 library for scalable asynchronous task execution on high-performance computing (HPC) systems. Integrated into a distributed traffic simulation workflow, ACE accelerates the computation of alternative routes, a key performance bottleneck in large-scale simulations. Unlike the previous Rust-based Evkit approach, ACE eliminates the multi-minute worker-spawning overhead and manages task granularity dynamically. Using scenarios for Prague and the Central Bohemia region, with datasets of up to 25 million routes, ACE achieved up to a 15x speed-up on city-scale workloads with shorter routes and a 1.45x improvement on larger regional workloads. These results highlight ACE's ability to adapt to workload characteristics and improve both efficiency and scalability in HPC-based route computation.

CCS Concepts

• **Computing methodologies** → *Simulation tools*; *Massively parallel algorithms*.

Keywords

High-Performance Computing, Traffic Simulation, Asynchronous Execution, C++17, Scalability, Performance Optimization

ACM Reference Format:
Paulo Silva, Pavlína Smolková, Kateřina Slaninová, Jan Martinovič, Matej Špeťko, João Barbosa, and Emanuele Vitali. 2025. Scalable Alternative Route Computation with ACE: A C++17 Library for HPC Traffic Simulations. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '25).* ACM, New York, NY, USA, 3 pages.

SC '25, St. Louis, MO, USA

1 Introduction

Efficient computation of alternative routes is a key requirement in large-scale traffic simulations, especially under dynamic traffic updates. The cost of computing alternatives increases with both the number of vehicles and the complexity of the road network.

In previous work [1], the RUTH traffic simulator [2] was integrated with Evkit [3], a Rust/C++ hybrid with a ZeroMQ backbone. Although functional, Evkit showed limitations for High Performance Computing (HPC) use: fixed batch size, static load balancing, multi-minute startup overhead (178–824 s) from worker spawning and toolchain compatibility issues that complicate deployment in different HPC clusters.

Asynchronous Communication and Execution (ACE) [1] was developed to address these issues. Implemented in C++17, it provides asynchronous task execution with pluggable communication back-ends (Message Passing Interface (MPI), Transmission Control Protocol (TCP)) and interoperability with Open Multi-Processing (OpenMP), Threading Building Blocks (TBB), and Compute Unified Device Architecture (CUDA). By dynamically adapting task granularity, ACE improves support for diverse workloads. In this work, we present the preliminary evaluation of ACE in two distinct scenarios in regions within the Czech Republic: Prague (city-like network, smaller graph) and Central Bohemia (larger graph with region-like network).

[1] https://opencode.it4i.eu/epicure/ace

2 ACE Library Overview

ACE is designed as a lightweight, self-contained library without external runtime dependencies. The core architecture comprises three components:

- Communication layer – an abstract interface with pluggable back-ends (MPI, TCP) to support different execution environments.
- Execution engine – a scheduler with worker threads and a dedicated coordinator thread, implementing dynamic load balancing through task donation.
- Task system – defines each workload unit via an execute, marshal, and size interface.

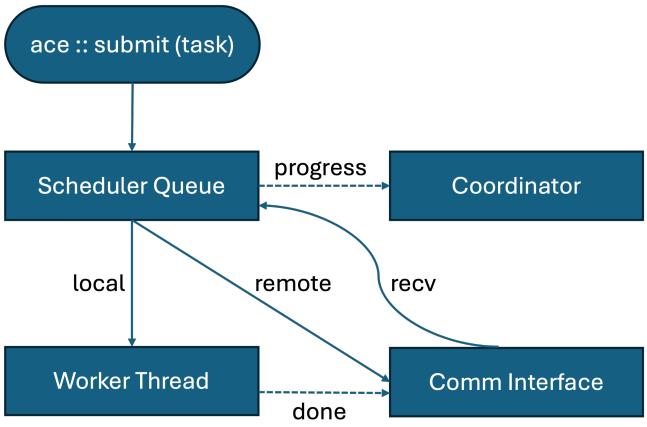

Figure 1: ACE workflow illustrating task submission, scheduling, execution, and communication.

Figure 1 depicts the conceptual workflow of ACE. The programming model is based on asynchronous tasks, with primitives for task submission (`ace::submit`), broadcasting (`ace::broadcast`), synchronization (`task->wait`), and coordinated termination (`ace::finalize`). ACE dynamically adapts the granularity of tasks, allowing the simulator to process both short alternative routes efficiently in compact networks and long computation-heavy routes in larger regional scenarios. In contrast, Evkit uses a fixed batch size, static scheduling, and a manual termination model, limiting scalability and flexibility. ACE's architecture enables efficient scaling on HPC systems while maintaining portability.

Table 1 summarizes the main differences between ACE and the previous Evkit-based approach.

3 Methodology

Two scenarios [2] with different geographic areas were considered for this evaluation:

- Central Bohemia Region – a large mixed urban–rural network where alternative routes are longer and contain more segments (Figure 2).
- Prague – a compact urban network where alternative routes are shorter and contain fewer road segments (Figure 3).

[2] https://doi.org/10.5281/zenodo.17206523

Table 1: EVKIT vs ACE Features

Feature	EVKIT	ACE
Language	Polyglot (Rust, C++, Py)	Pure C++17
Backbone	ZeroMQ	MPI / TCP
Architecture	Broker	Distributed Work-Stealing
Asynch Model	Manual Threading	Cooperative Tasking
Load Balancing	Static (Round-robin)	Dynamic (Task Donation)
Termination	Manual / Ad-hoc	Automatic (Coordinator)
Scalability	Limited	Massively Parallel
Collectives	Not Supported	Barriers, Broadcast

In both cases, Origin–Destination matrices define the number and distribution of routes. The computational workload is determined by the number of alternative routes calculated, with the cost depending on the path length and the number of edges in the network graph.

The alternative route computations were executed with ACE replacing Evkit as the distributed execution layer. Unlike the previous setup that required explicit tuning of workload and batch sizes, ACE now manages task granularity dynamically through its runtime scheduler. This allows the system to adapt automatically to the different workload characteristics of the two scenarios: in Prague (shorter alternative routes), the scheduler minimizes latency, while in the larger Central Bohemia region (longer alternatives), it balances communication and computation to maintain efficiency.

All experiments were performed on Karolina supercomputer (IT4Innovations, Czech Republic). The metrics collected include computation time, startup overhead, and scalability as a function of node count, evaluated from 1 to 16 nodes. Analysis with higher node count and on different clusters is part of ongoing work.

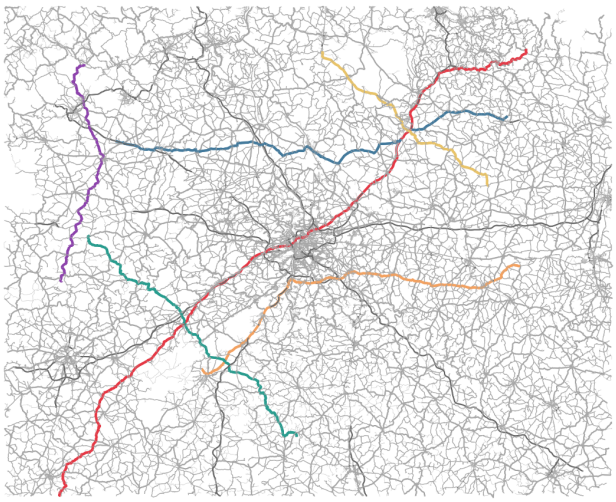

Figure 2: Central Bohemia Region road network with sample routes (Czech Republic).

Figure 3: Traffic simulation in Prague showing segment speeds at a given time (Czech Republic).

4 Results

Figure 4 shows the speed-up trend for Prague (2.6M routes), where ACE achieves up to 15× improvement. Table 2 provides absolute times, separating Evkit compute and spawn overhead. For the larger Central Bohemia case (25M routes), shown in Figure 5, ACE delivers a 1.45× improvement. The analysis is similar, with Evkit's spawn overhead contributing significantly but being dimmed by longer compute times.

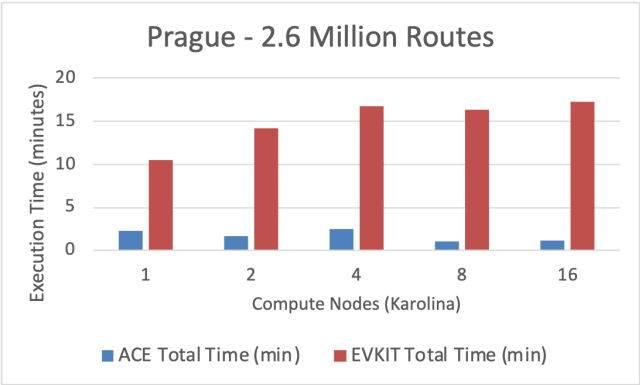

Figure 4: Computation of 2.6M routes in Prague, evaluated from 1 to 16 nodes.

5 Conclusion

ACE provides a flexible and efficient task execution layer for HPC traffic simulation. Compared to the previous Evkit-based approach, ACE currently eliminates multi-minute startup overhead and scales effectively up to 8 nodes. Experiments show up to 15x speed-up on city-scale datasets and 1.45x on regional-scale workloads, reflecting the influence of route length and dataset size. The ongoing work focuses on mitigating scaling saturation beyond 16 nodes and further improving performance on longer routes. The design also provides flexibility to potentially adapt ACE to other distributed, task-based HPC applications.

Table 2: Execution times for Prague scenario (2.6M routes, 1–16 nodes).

Nodes	ACE Total (min)	Evkit Total (min)	Evkit Comp. Time (min)	Evkit Spawn Time (min)
1	2.3	10.5	7.6	3.0
2	1.7	14.2	6.9	7.4
4	2.5	16.7	6.9	9.9
8	1.1	16.3	6.8	9.5
16	1.2	17.3	6.8	10.5

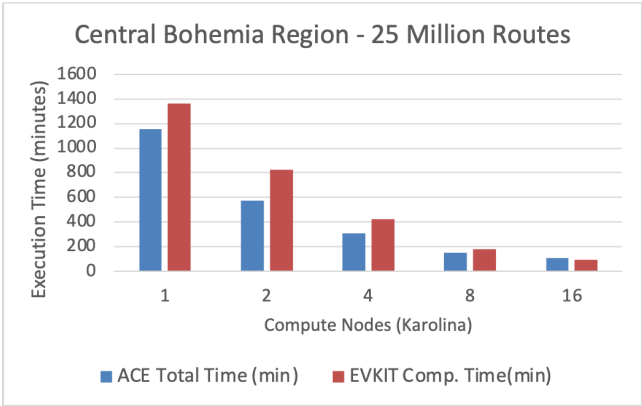

Figure 5: Computation of 25M routes in the Central Bohemia region, evaluated from 1 to 16 nodes.

Acknowledgments

This work was also supported by the EPICURE project under grant agreement No 101139786. The project is supported by the European High-Performance Computing Joint Undertaking and its members (including top-up funding by the Ministry of Education, Youth and Sports of the Czech Republic ID: MC2402).

References

[1] Paulo Silva, Pavlína Smolková, et al. Assessing the impact of real-time traffic updates on traffic flow: A high-performance computing perspective on scalability and demand. Poster presented at SC24: The International Conference for High Performance Computing, Networking, Storage, and Analysis, November 2024. Research Poster and ACM Student Research Competition Poster Archive.

[2] Paulo Silva, Pavlína Smolková, Sofia Michailidu, Jakub Beránek, Roman Macháček, Kateřina Slaninová, Jan Martinovič, and Radim Cmar. High-performance computing for distributed route computation in traffic flow models. *Procedia Computer Science*, 255:83–92, 2025. Proceedings of the Second EuroHPC user day.

[3] Christian Pilato, Subhadeep Banik, Jakub Beránek, et al. A system development kit for big data applications on fpga-based clusters: The everest approach. In *Proceedings of the 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2024.