Ozone: Fully Out-of-Order Choreographies (Artifact)

Dan Plyukhin
University of Southern Denmark, Odense, Denmark

Marco Peressotti
University of Southern Denmark, Odense, Denmark

Fabrizio Montesi
University of Southern Denmark, Odense, Denmark

Abstract

Choreographic programming is a paradigm for writing distributed applications. It allows programmers to write a single program, called a choreography, that can be compiled to generate correct implementations of each process in the application. Although choreographies provide good static guarantees, they can exhibit high latency when messages or processes are delayed. This is because processes in a choreography typically execute in a fixed, deterministic order, and cannot adapt to the order that messages arrive at runtime. In non-choreographic code, programmers can address this problem by allowing processes to execute out of order – for instance by using futures or reactive programming. However, in choreographic code, out-of-order process execution can lead to serious and subtle bugs, called *communication integrity violations (CIVs)*.

In this paper, we develop a model of choreographic programming for out-of-order processes that guarantees absence of CIVs and deadlocks. As an application of our approach, we also introduce an API for safe non-blocking communication via futures in the choreographic programming language Choral. The API allows processes to execute out of order, participate in multiple choreographies concurrently, and to handle unordered data messages. We provide an illustrative evaluation of our API, showing that out-of-order execution can reduce latency and increase throughput by overlapping communication with computation.

2012 ACM Subject Classification Computing methodologies → Concurrent computing methodologies

Keywords and phrases Choreographic programming, Asynchrony, Concurrency.

Digital Object Identifier 10.4230/DARTS.10.2.16

Funding Partially supported by Villum Fonden (grant no. 29518). Co-funded by the European Union (ERC, CHORDS, 101124225). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Related Article Dan Plyukhin, Marco Peressotti, and Fabrizio Montesi, "Ozone: Fully Out-of-Order Choreographies", in 38th European Conference on Object-Oriented Programming (ECOOP 2024), LIPIcs, Vol. 313, pp. 31:1–31:28, 2024.
`https://doi.org/10.4230/LIPIcs.ECOOP.2024.31`

Related Conference 38th European Conference on Object-Oriented Programming (ECOOP 2024), September 16–20, 2024, Vienna, Austria

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2024 Call for Artifacts and the ACM Artifact Review and Badging Policy.

1 Scope

The artifact validates the experimental results found in the paper. Namely, we ran two microbenchmarks and a model serving benchmark on a six-node Linux cluster. We found that Ozone could achieve significantly better scalability than vanilla Choral under high request loads, and could achieve better fairness when a server interacts with multiple clients concurrently.

2 Content

The artifact package includes:

- Source code for the Ozone library
- Raw data from benchmarks
- Scripts for processing and plotting raw data
- Figures included in the paper

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://github.com/dplyukhin/ozone`.

4 Tested platforms

The artifact is known to work on MacOS 14.4 and Ubuntu 24.04.

5 License

The artifact is available under license GPL 2.0.

6 MD5 sum of the artifact

047bad286b879821a76d928d62653811

7 Size of the artifact

2.4MB