ConDRust: Scalable Deterministic Concurrency from Verifiable Rust Programs (Artifact)

Felix Suchert
TU Dresden, Germany

Lisza Zeidler
Barkhausen Insitut, Dresden, Germany

Jeronimo Castrillon
TU Dresden, Germany

Sebastian Ertel
Barkhausen Institut, Dresden, Germany

Abstract

SAT/SMT-solvers and model checkers automate formal verification of sequential programs. Formal reasoning about scalable concurrent programs is still manual and requires expert knowledge. But scalability is a fundamental requirement of current and future programs.

Sequential imperative programs compose statements, function/method calls and control flow constructs. Concurrent programming models provide constructs for concurrent composition. Concurrency abstractions such as threads and synchronization primitives such as locks compose the individual parts of a concurrent program that are meant to execute in parallel. We propose to rather compose the individual parts again using sequential composition and compile this sequential composition into a concurrent one. The developer can use existing tools to formally verify the sequential program while the translated concurrent program provides the dearly requested scalability.

Following this insight, we present *ConDRust*, a new programming model and compiler for Rust programs. The *ConDRust* compiler translates sequential composition into a concurrent composition based on threads and message-passing channels. During compilation, the compiler preserves the semantics of the sequential program along with much desired properties such as determinism.

Our evaluation shows that our *ConDRust* compiler generates concurrent deterministic code that can outperform even non-deterministic programs by up to a factor of three for irregular algorithms that are particularly hard to parallelize.

2012 ACM Subject Classification Theory of computation → Parallel computing models; Software and its engineering → Parallel programming languages

Keywords and phrases concurrent programming, verification, scalability

Digital Object Identifier 10.4230/DARTS.9.2.16

Funding *Felix Suchert*: was funded by the EU Horizon 2020 Programme under grant agreement No 957269 (EVEREST).
Lisza Zeidler: was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 469256231.

Related Article Felix Suchert, Lisza Zeidler, Jeronimo Castrillon, and Sebastian Ertel, "ConDRust: Scalable Deterministic Concurrency from Verifiable Rust Programs", in 37th European Conference on Object-Oriented Programming (ECOOP 2023), LIPIcs, Vol. 263, pp. 33:1–33:39, 2023.
`https://doi.org/10.4230/LIPIcs.ECOOP.2023.33`

Related Conference 37th European Conference on Object-Oriented Programming (ECOOP 2023), July 17–21, 2023, Seattle, Washington, United States

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2023 Call for Artifacts and the ACM Artifact Review and Badging Policy.

Dagstuhl Artifacts Series, Vol. 9, Issue 2, Artifact No. 16, pp. 16:1–16:3
DAGSTUHL ARTIFACTS SERIES
Dagstuhl Artifacts Series
Schloss Dagstuhl – Leibniz-Zentrum für Informatik,
Dagstuhl Publishing, Germany

1 Scope

This artifact aims to back the performance claims made in the accompanying paper. It contains our presented implementations of various benchmarks from the STAMP [3], PARSEC [1] and YCSB [2] suites. We also include the build of the *ConDRust* compiler (named `ohuac`) used to generate the code for the benchmarks as well as all external libraries used.

Additionally, this repository contains the proof that an explicit panic we inserted in the code as part of our optimization is never encountered in the sequential version of the code.

2 Content

The artifact package is a Docker image which includes:

- The *ConDRust* compiler, named `ohuac` (binary)
  Source code available on `https://github.com/ohua-lang/condrust`.
- A patched version of the rust-stm library fixing a deadlock problem (code)
  Original available on `https://github.com/Marthog/rust-stm`.
- A library providing rudimentary STM-aware data structures (code)
- Performance benchmarks for various benchmarks from STAMP and PARSEC that appeared in our paper (benchmark)
- A Key-Value Store implementation also appearing in our paper (benchmark)
- A correctness proof for the absence of panics in an optimization we perform (proof)

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).

4 Tested platforms

The artifact is provided as Docker file and should therefore run on all platforms supported by the software. It is known to run on Ubuntu 22.04, NixOS 22.11 and macOS 13.

5 License

The artifact is available under Creative Commons Attribution 4.0 International license (CC BY 4.0).

6 MD5 sum of the artifact

670b7c3d47310b956acf3c451b35a6a9

7 Size of the artifact

1.7 GiB

References

1 Christian Bienia and Kai Li. Parsec 2.0: A new benchmark suite for chip-multiprocessors. In *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, volume 2011, page 37, 2009.

2 Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154, 2010.

3 Chi Cao Minh, JaeWoong Chung, Christos Kozyrakis, and Kunle Olukotun. Stamp: Stanford transactional applications for multi-processing. In *2008 IEEE International Symposium on Workload Characterization*, pages 35–46. IEEE, 2008.