


Compiling Volatile Correctly in Java (Artifact)

Shuyang Liu ✉ 

University of California, Los Angeles, CA, USA

John Bender ✉

Sandia National Laboratories, Albuquerque, NM, USA

Jens Palsberg ✉

University of California, Los Angeles, CA, USA

— Abstract —

The compilation scheme for `Volatile` accesses in the OpenJDK 9 HotSpot Java Virtual Machine has a major problem that persists despite a recent bug report and a long discussion. One of the suggested fixes is to let Java compile `Volatile` accesses in the same way as `C/C++11`. However, we show that this approach is invalid for Java. Indeed, we show a set of optimizations that is valid for `C/C++11` but

invalid for Java, while the compilation scheme is similar. We prove the correctness of the compilation scheme to Power and x86 and a suite of valid optimizations in Java. Our proofs are based on a language model that we validate by proving key properties such as the DRF-SC theorem and by running litmus tests via our implementation of Java in Herd7.

2012 ACM Subject Classification Software and its engineering → Semantics

Keywords and phrases formal semantics, concurrency, compilation

Digital Object Identifier 10.4230/DARTS.8.2.3

Funding This material is based upon work supported by the National Science Foundation under Grant No. 1815496.

Acknowledgements We thank Doug Lea for the helpful insights on the Java language semantics and compilers; we thank Jade Alglave for her precious and detailed help on implementing Java architecture for Herd; we thank Ori Lahav, Anton Podkopaev and Viktor Vafeiadis for initially pointing out the issue of the Java Access Modes model; we thank all the reviewers of ECOOP'22 for their insightful feedback.

Related Article Shuyang Liu, John Bender, and Jens Palsberg, “Compiling Volatile Correctly in Java”, in 36th European Conference on Object-Oriented Programming (ECOOP 2022), LIPIcs, Vol. 222, pp. 6:1–6:26, 2022.

<https://doi.org/10.4230/LIPIcs.ECOOP.2022.6>

Related Conference 36th European Conference on Object-Oriented Programming (ECOOP 2022), June 6–10, 2022, Berlin, Germany

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2022 Call for Artifacts and the ACM Artifact Review and Badging Policy.

1 Scope

This artifact includes three parts:

- The extended Herd7 [1] implementation with Java architecture (code): the extended herd7 with support for the Java `VarHandle` syntax. The code should build with no problem and is able to run litmus tests written in the Java `VarHandle` syntax.
- Litmus Tests that appeared in our paper (benchmark): we use the benchmark litmus tests to show that our fixed memory model behave the same as the original memory model except for the problematic cases. For the problematic cases, we show that our fixed memory model solve the original problem.
- Coq proofs for some of the theorems in the paper (proof): the proofs of some theorems in the paper are sound (compiles through coq). This includes the theorems about properties of JAM_{21} .



© Shuyang Liu, John Bender, and Jens Palsberg;
licensed under Creative Commons License CC-BY 4.0
Dagstuhl Artifacts Series, Vol. 8, Issue 2, Artifact No. 3, pp. 3:1–3:2



DAGSTUHL
ARTIFACTS SERIES

Dagstuhl Artifacts Series
Schloss Dagstuhl – Leibniz-Zentrum für Informatik,
Dagstuhl Publishing, Germany



3:2 A Sample DARTS Research Description (Artifact)

2 Content

The artifact package includes:

- The extended Herd7 implementation with Java architecture (code)
- Litmus Tests that appeared in our paper (benchmark)
- Coq proofs for some of the theorems in the paper (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). In addition, the artifact is also available at: <https://github.com/ShuyangLiu/ECOOP22-Supplementary-Material>.

4 Tested platforms

- The extended Herd7 implementation with Java architecture (code): Requires Ocaml 4.09.0 and dune 2.9.1
- Litmus Tests that appeared in our paper (benchmark): Can be executed using the extended Herd7 tool with the JAM model
- Coq proofs for some of the theorems in the paper (proof): Requires Coq 8.06.1 with Ocaml 4.02.3

5 License

The artifact is available under license Creative Commons license.

6 MD5 sum of the artifact

68b11fb27dbd97fbe5d9588b133d9658

7 Size of the artifact

1.5M

References

- 1 Jade Alglave, Luc Maranget, and Michael Tautschnig. Herding cats: Modelling, simulation, testing, and data mining for weak memory. *ACM Trans. Program. Lang. Syst.*, 36(2), July 2014. doi:10.1145/2627752.