Automated Large-Scale Multi-Language Dynamic Program Analysis in the Wild (Artifact)

Alex Villazón
Universidad Privada Boliviana, Bolivia
avillazon@upb.edu

Haiyang Sun
Università della Svizzera italiana, Switzerland
haiyang.sun@usi.ch

Andrea Rosà
Università della Svizzera italiana, Switzerland
andrea.rosa@usi.ch

Eduardo Rosales
Università della Svizzera italiana, Switzerland
rosale@usi.ch

Daniele Bonetta
Oracle Labs, United States
daniele.bonetta@oracle.com

Isabella Defilippis
Universidad Privada Boliviana, Bolivia
isabelladefilippis@upb.edu

Sergio Oporto
Universidad Privada Boliviana, Bolivia
sergiooporto@upb.edu

Walter Binder
Università della Svizzera italiana, Switzerland
walter.binder@usi.ch

Abstract

This artifact provides a preliminary release of NAB, a distributed infrastructure for executing large-scale dynamic program analyses (DPAs). The artifact consists of ready-to-use Docker containers that allow one to run different DPA tools (Deep-Promise, JITProf, and tgp) on Node.js, Java, and Scala projects hosted on GitHub. The artifact enables the reproduction of the figures and tables of the related paper "Automated Large-scale Multi-language Dynamic Program Analysis in the Wild" with pre-collected data (several GBs) and the execution of DPAs on specific sets of GitHub projects.

2012 ACM Subject Classification Software and its engineering → Dynamic analysis

Keywords and phrases Dynamic program analysis, code repositories, GitHub, Node.js, Java, Scala, promises, JIT-unfriendly code, task granularity

Digital Object Identifier 10.4230/DARTS.5.2.11

Acknowledgements This work has been supported by Oracle (ERO project 1332), Swiss National Science Foundation (scientific exchange project IZSEZ0_177215), Hasler Foundation (project 18012), and by a Bridging Grant with Japan (BG 04-122017).

Related Article Alex Villazón, Haiyang Sun, Andrea Rosà, Eduardo Rosales, Daniele Bonetta, Isabella Defilippis, Sergio Oporto, and Walter Binder, "Automated Large-Scale Multi-Language Dynamic Program Analysis in the Wild", in 33rd European Conference on Object-Oriented Programming (ECOOP 2019), LIPIcs, Vol. 134, pp. 20:1–20:27, 2019. https://dx.doi.org/10.4230/LIPIcs.ECOOP.2019.20

Related Conference 33rd European Conference on Object-Oriented Programming (ECOOP 2019), July 15–19, 2019, London, United Kingdom

1 Scope

The artifact contains a preliminary release of NAB, including different services (i.e., NAB-Master, NAB-Crawler, NAB-Analyzer, and NAB-Dashboard) that run in Docker containers, as described in the related paper "Automated Large-scale Multi-language Dynamic Program Analysis in the Wild" [6]. The NAB-Analyzer Docker image in the artifact includes three DPA tools (DeepPromise [6], JITProf [2] and tgp [5, 3, 4] and two runtimes (Oracle's JVM [1] and GraalVM [7]) for analyzing open-source Java, Scala, and Node.js projects hosted on GitHub.

To reproduce the figures and tables from the related paper, we use pre-collected data (i.e., results of DPA tools applied to more than 56K Node.js, Java, and Scala projects from GitHub) stored in MongoDB databases. The pre-collected data is stored in the NAB-Dashboard image. We provide scripts to restore the database, to process the results, and to generate the tables and figures.

The artifact also includes scripts to execute the DPA tools on the full set of GitHub projects used to generate the pre-collected data (i.e., to reproduce the experiments), as well as to execute the DPA tools on smaller subsets of selected GitHub projects (for evaluating NAB on a smaller input set). The scripts automate different activities, such as cloning, building, running test code applying the selected DPA tool, collecting DPA results, and generating output in the form of tables and figures.

2 Content

The artifact includes the following content:

- `docker/`: directory containing four Docker images, one for each NAB core service (`nab-master.img`, `nab-analyzer.img`, `nab-crawler.img`, and `nab-dashboard.img`).
- `nab/`: directory containing scripts, configuration files for each case study, sample configuration files, and NAB service deployment files.
- `utils/`: directory containing helper scripts.
- `load.sh`: script to load the images of NAB services in the Docker installation.
- `README.pdf`: a description about how to run the Docker images containing NAB services for the three DPAs covered in the related paper.
- `run-nab.sh`: the main script to start the core NAB services and run the experiments.

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: `http://research.upb.edu/NAB/nab-artifact.tgz`.

4 Tested platforms

The artifact has been successfully tested on machines running Linux or macOS with at least 16 GB of RAM and 50 GB of free storage. It was tested with Docker 18.06.1-ce (build e68fc7a) and 18.09.2 (build 6247962).

5 License

The artifact is available under Apache License 2.0.

6 MD5 sum of the artifact

fa2151a3fa429d676e7eb2c77d5aad62

7 Size of the artifact

5.6 GB

References

1 Oracle Corporation. Java SE HotSpot at a Glance. `https://www.oracle.com/technetwork/java/javase/tech/index-jsp-136373.html`, 2018.

2 L. Gong, M. Pradel, and K. Sen. JITProf: Pinpointing JIT-unfriendly JavaScript Code. In *ESEC/FSE*, pages 357–368, 2015.

3 A. Rosà, E. Rosales, and W. Binder. Analyzing and Optimizing Task Granularity on the JVM. In *CGO*, pages 27–37, 2018.

4 A. Rosà, E. Rosales, and W. Binder. Analysis and Optimization of Task Granularity on the Java Virtual Machine. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, pages 1–47, 2019.

5 E. Rosales, A. Rosà, and W. Binder. tgp: a Task-Granularity Profiler for the Java Virtual Machine. In *APSEC*, pages 570–575, 2017.

6 A. Villazón, H. Sun, A. Rosà, E. Rosales, D. Bonetta, I. Defilippis, S. Oporto, and W. Binder. Automated Large-scale Multi-language Dynamic Program Analysis in the Wild. In *ECOOP*, London, UK, July 2019.

7 T. Würthinger, C. Wimmer, C. Humer, A. Wöß, L. Stadler, C. Seaton, G. Duboscq, D. Simon, and M. Grimmer. Practical Partial Evaluation for High-performance Dynamic Language Runtimes. In *PLDI*, pages 662–676, 2017.