Magic with Dynamo – Flexible Cross-Component Linking for Java with Invokedynamic (Artifact)*

Kamil Jezek[1] and Jens Dietrich[2]

1 **NTIS – New Technologies for the Information Society**
Faculty of Applied Sciences, University of West Bohemia
Pilsen, Czech Republic
kjezek@kiv.zcu.cz

2 **School of Engineering and Advanced Technology, Massey University**
Palmerston North, New Zealand
j.b.dietrich@massey.ac.nz

Abstract

Modern software systems are not built from scratch. They use functionality provided by libraries. These libraries evolve and often upgrades are deployed without the systems being recompiled. In Java, this process is particularly error-prone due to the mismatch between source and binary compatibility, and the lack of API stability in many popular libraries. We propose a novel approach to mitigate this problem based on the use of `invokedynamic` instructions for cross-component method invocations. The dispatch mechanism of `invokedynamic` is used to provide on-the-fly signature adaptation. We show how this idea can be used to construct a Java compiler that produces more resilient bytecode. We present the dynamo compiler, a proof-of-concept implemented as `javac` post compiler, and evaluate our approach using several benchmark examples and two case studies showing how the use of the dynamo compiler can prevent certain types of linkage and stack overflow errors that have been observed in real-world systems.

1998 ACM Subject Classification D.1.5 Object-oriented Programming, D.2.13 Reusable Software, D.3.4 Processors

Keywords and phrases Java, compilation, linking, binary compatibility, invokedynamic

Digital Object Identifier 10.4230/DARTS.2.1.5

Related Article Kamil Jezek and Jens Dietrich, "Magic with Dynamo – Flexible Cross-Component Linking for Java with Invokedynamic", in Proceedings of the 30th European Conference on Object-Oriented Programming (ECOOP 2016), LIPIcs, Vol. 56, pp. 12:1–12:25, 2016.
http://dx.doi.org/10.4230/LIPIcs.ECOOP.2016.12

Related Conference 30th European Conference on Object-Oriented Programming (ECOOP 2016), July 18–22, 2016, Rome, Italy

1 Scope

The artifact is designed to support repeatability of all the experiments of the companion paper, allowing users to test *dynamo* on a variety of benchmarks.

2 Content

The artifact is packaged as a VM image that can be opened with VirtualBox version 4.2.30 or better. VirtualBox can be obtained for free from `https://www.virtualbox.org/wiki/Downloads`. The

* This work was partially supported by Oracle Labs, Australia.

VM uses Ubuntu 15.10, and has several tools (Java, Maven, Ant) pre-installed. It also contains the complete project source code, and pre-built binaries.

Once the VM started and the user has logged in, the script *run-evaluation.sh* located on the desktop (*∼/Desktop*) can be used to run a "guided" evaluation. It builds *dynamo* and runs all experiments. Between steps a short description of the work done / to be done is displayed and the user is prompted for input to continue. The whole project is stored in *∼/dynamo*.

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). The latest version of our code is also available from the bitbucket project web site: `https://bitbucket.org/kjezek/dynamo`.

4 Tested platforms

The artifact is known to work on any platform running Oracle VirtualBox version 4 (`https://www.virtualbox.org/`). To get meaningful results from experiments, it is recommended to use at least 4GB of RAM and 2 CPU cores. Otherwise performance-related results may be compromised by swapping and Java garbage collector activity.

5 License

The MIT License (`https://opensource.org/licenses/MIT`)

6 MD5 sum of the artifact

6f769a041059f704068b5a9e4b387c04

7 Size of the artifact

2.4 GB