JOHANNES KEPLER
UNIVERSITY LINZ

o.Univ.-Prof. Dr. Dr.h.c.
Hanspeter Mössenböck
Institut für Systemsoftware

P +43 732 2468 4340
F +43 732 2468 4345
hanspeter.moessenboeck@jku.at

Sekretariat:
Karin Gusenbauer
Ext. 4342
karin.gusenbauer@jku.at

Master's Thesis

Advanced Debugging of Native Images in GDB

Student: Dominik Mascherbauer (12005607)
Advisor: Prof. Dr. Hanspeter Mössenböck
Co-advisors: Paul Wögerer, Christian Wirth

Start: 01.03.2024

Linz, 1. March 2024

GraalVM [1] Native Image [2] provides ahead-of-time compilation of Java code to optimized native binaries that run without a bundled JDK. As it generates native code, debugging such applications is predominantly done using tools (such as GDB) developed for native code.

GDB provides an API [3] that allows one to improve debugging of such applications by implementing a Python script that controls and enriches the debugging experience for the user. The student's master project “Debugging of Native Images using a Python Layer in GDB” successfully completed the creation of such a script. As part of this Master's Thesis, the implementation should be enhanced further to cover more native debugging use cases.

While most use cases of debugging images solely rely on debuginfo generated at image build time there are also more advanced use cases. Native Image can also generate images that contain Truffle languages with runtime compilation support [4]. These images perform code generation at image-runtime. Debuginfo generation at image build-time is insufficient to provide all debuginfo needed to debug such images.

This problem needs to be solved by allowing debuginfo generation also at image-runtime. When a new method gets compiled at run time, debuginfo generation needs to be performed as well. After completion, the debugger needs to be informed that new debuginfo is available in the current debug session. This can be communicated to GDB via its JIT Compilation Interface [5].

As part of this master thesis the existing code-base for debuginfo generation needs to be retrofitted to also be suitable for inclusion into Native-Image VM-level code so that it can be used at image-runtime. This requires footprint reduction, i.e., debuginfo generation is only allowed to rely on

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at
DVR 0093696

parts of the JDK that are light-weight and do not significantly increase the size of images that make use of runtime compilation.

A common shared base needs to be established that can be used for both, debuginfo generation at built time and run time. Based on that, there have to be the two concrete implementations. The hosted implementation that corresponds to the existing implementation and the new run-time variant. The latter needs to output debuginfo into native memory instead of to a file and also only output the delta of debuginfo to the debuginfo already created at image build time. Subsequently the run-time variant needs to be integrated into the Graal compilation

and code installation pipeline so that it gets informed whenever a new native method is available. Finally, the interfacing with GDB needs to be implemented to ensure that GDB gets notified whenever new debuginfo gets available or when debuginfo needs to be revoked again in case of deoptimization of run-time-compiled code.

The scope of this thesis is as follows:

- Provide high-level views for Java and Graal Collections (List Set Map) as well as for EconomicSet/EconomicMap and provide testing of those high-level views.
- Ensure that GDB debugging works well with native-image shared libraries (e.g. debugging of libgraal that runs in HotSpot, support classloader-prefixes in classnames)
- Implement debuginfo generation at image-runtime as described above.
- Make sure that gdb backtrace works during deoptimization of run-time-compiled code (created by Truffle PE)
- Test the approach on reasonable applications, and provide, where applicable, unit tests and a testing harness.
- Contribute the approach to the repository of GraalVM [2] (might require signing OCA [6]).

The work's progress should be discussed with the supervisor at least every 2 weeks. Please note the guidelines of the Institute for System Software when preparing the written thesis. The deadline for the written thesis is February 28th, 2025.

References:

[1] https://www.github.com/oracle/graal
[2] https://github.com/oracle/graal/tree/master/substratevm
[3] https://sourceware.org/gdb/onlinedocs/gdb/Python-API.html
[4] http://lafo.ssw.uni-linz.ac.at/papers/2013_Onward_OneVMToRuleThemAll.pdf
[5] https://sourceware.org/gdb/current/onlinedocs/gdb.html/JIT-Interface.html
[6] https://oca.opensource.oracle.com/

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Str. 69
4040 Linz, Austria
jku.at
DVR 0093696