Interoperability between Execution Modes on GraalVM

Christoph Pichler
Johannes Kepler University
Linz, Austria
christoph.pichler@jku.at

Paley Li
Oracle Labs
Prague, Czech Republic
paley.li@oracle.com

Roland Schatz
Oracle Labs
Linz, Austria
roland.schatz@oracle.com

Hanspeter Mössenböck
Johannes Kepler University
Linz, Austria
hanspeter.moessenboeck@jku.at

ABSTRACT

GraalVM[1] [2] is a virtual machine, whose core is an optimizing JIT-compiler [1]. Its polyglot execution environment supports a variety of programming languages, such as Java, JavaScript, Python, Ruby and C. As all code is run in a single runtime, this also enables cross-language function calls. To run C code in GraalVM, first it has to be compiled to LLVM[2] bitcode, which is then interpreted and executed in GraalVM ("managed execution"). However, GraalVM can also invoke native code that is executed on the bare machine via its native function interface. Thus, compiling the C code to a (binary) executable and using the native function interface is another option ("native execution"). The goal of this thesis is to find heuristics for deciding when such code should be executed in native or in managed mode as well as mechanisms for mixed-mode execution.

Currently, GraalVM prefers managed execution whenever LLVM bitcode is available. However, this leads to elevated start-up times due to class loading and warm-up. We argue that more native execution within GraalVM leads to a better performance, but it also comes with certain limitations.

As an example, consider Python code running on GraalVM – more precisely, passing GraalVM data to a function of the Python library PyTorch[3], which is implemented in C: Although only a minority of instructions in PyTorch might access GraalVM data and thus has to be run in GraalVM, currently the whole library has to be run in GraalVM. Ideally, the majority of the code will be run natively, leading to a higher performance.

In our work, we propose and implement a hybrid mode for GraalVM's LLVM component: Each function can either be run in native mode or in GraalVM. Code which needs features of GraalVM is executed in managed mode, while the remaining functions are executed natively and do not suffer from long start-up times. However, some challenges and restrictions still have to be considered:

- Program states have to be kept consistent: For state information, e.g. for a global variable, its value during native execution must be consistent with its value during managed execution. It is neither possible to have all variables stored only in native mode, nor all of them only in managed mode.
- Moreover, it cannot be determined by static code analysis if managed data originating from GraalVM, e.g. objects from foreign languages, are accessed only in managed mode during execution. Therefore, it can happen that managed data are about to be accessed in native execution. In such case, it is necessary to switch the execution mode at run time.
- Another important aspect is how the decision between native and managed execution should be taken: Currently, the developer has to choose the mode (native/managed) for each function. However, we aim for a scenario where the mode is automatically chosen from dynamic profiling data and other analysis.

CCS CONCEPTS

• **Software and its engineering** → **Runtime environments**; **Just-in-time compilers**; **Software performance**.

KEYWORDS

Virtual machine, JIT-compilation, Warm-up performance, Native execution, Managed execution

ACM Reference Format:
Christoph Pichler, Paley Li, Roland Schatz, and Hanspeter Mössenböck. 2022. Interoperability between Execution Modes on GraalVM. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Student: Christoph Pichler
Advisors: Paley Li, Roland Schatz, Hanspeter Mössenböck

REFERENCES

[1] John Aycock. 2003. A Brief History of Just-in-Time. *ACM Comput. Surv.* 35, 2 (jun 2003), 97–113. https://doi.org/10.1145/857076.857077
[2] Thomas Würthinger, Christian Wimmer, Andreas Wöß, Lukas Stadler, Gilles Duboscq, Christian Humer, Georg Richards, Doug Simon, and Mario Wolczko. 2013. One VM to rule them all. In *Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming and software* (Indianapolis, Indiana, USA). ACM, New York, NY, USA, 187–204. https://doi.org/10.1145/2509578.2509581

[1] https://www.graalvm.org/
[2] https://llvm.org/
[3] https://pytorch.org/

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn