LLVM vs JITBuilder: Measuring and Comparing the Overhead of two API-based JIT Frameworks

Eric Coffin
Faculty of Computer Science
University of New Brunswick
Fredericton, NB, Canada
eric.coffin@unb.ca

ABSTRACT

Just-in-Time compilation has allowed for significant performance gains during the run-time of applications. LLVM can be embedded within an application to allow JIT compilation during run-time. Similarly, the OMR JIT compiler can be embedded within an application. Both frameworks offer simple, programmer interaces to define and generate native methods at run-time. In this report we discuss the different approaches the two frameworks employ. We then measure the overhead associated with each framework while compiling relatively simple functions. We found that while LLVM required a larger memory footprint, in certain cases it was able to generate code more quickly. Furthermore, in our tests, the code LLVM generated typically offered high throughput. JitBuilder, a relatively young project compared to LLVM, does not expose the underlying configuration of TRJIT, instead locking compilations in to the *warm* optimization level.

CCS CONCEPTS

• **Software and its engineering → Compilers**.

KEYWORDS

compilers, just-in-time, optimization

7 ACKNOWLEDGEMENTS

This research was conducted within the Centre for Advanced Studies—Atlantic, Faculty of Computer Science, University of New Brunswick. The authors are grateful for the colleagues and facilities of CAS Atlantic in supporting our research. The authors would like to acknowledge the funding support provided by the Atlantic Canada Opportunities Agency (ACOA) through the Atlantic Innovation Fund (AIF) program. Furthermore, we would also like to thank the New Brunswick Innovation Foundation for contributing to this project.

REFERENCES

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