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 significantly larger memory footprint, it was able to generate code more quickly. Furthermore, the code LLVM generated was by default more optimized. By configuring JITBuilder, we were able to reduce the compilation time significantly, however it still was above that of LLVM.

CCS CONCEPTS

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

KEYWORDS

compilers, just-in-time, optimization

1 INTRODUCTION

Introduction

2 BACKGROUND

2.1 LLVM

LLVM.

2.2 JITBuilder

JITBuilder.

3 METHODOLOGY

Methodology

3.1 Results

4 RELATED WORK

Related work.

5 FUTURE WORK

Future Work

6 SUMMARY

Summary

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