Bachelor Thesis

High performance FFI in Julia: A case study of the FFI landscape

by

Kai Erik Niermann

(2720905)

First supervisor: Atze Van der Ploeg
Daily supervisor: Atze Van der Ploeg
Second reader: Verano Merino

June 30, 2024

Submitted in partial fulfillment of the requirements for
the VU degree of Bachelor of Science in Computer Science

High performance FFI in Julia: A case study of the FFI landscape

Kai Erik Niermann
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
k.e.niermann@student.vu.nl

ABSTRACT

Your Master Thesis has to be written in English and should follow the ACM style (see the Overleaf template). We also suggest a *structured abstract* following the structure below.
Context. Write this at the end
Goal. Write this at the end
Method. Write this at the end
Results. Write this at the end
Conclusions. Write this at the end

As a very rough indication, the final thesis report typically entails, excluding appendixes, between 10 and 30 pages, with an average of 15 pages. The large variation depends on many variables including the specific field, project nature, and context. We advise to ask your supervisor if you should consider which number as a reference.

1 INTRODUCTION

Rewriting software components; or entire systems; in a different language is a common practice to address various limitations caused by the original implementation language, including but not limited to performance, maintainability, and scalability. The primary motivation behind rewriting is to leverage the strengths of the new language to address the limitations of the original language. However, rewriting a large system, especially in a different language with potentially different paradigms can be a complex and time-consuming task. This study aims to asses the viability of leveraging language interoperability to simplify and expedite the rewriting process.

Most well known major software services today like Facebook, YouTube, Google, LinkedIn, Wordpress, etc. have all had either components many of them starting out with language such as PHP or Ruby on Rails because these were the popular choices back in the day and what the intial developers where likely the most familiar with. We can take YouTube as a good example, it was intially written largely in PHP, but as the platform grew this came with some key drawbacks, a key one being scalability, one reason for switching to Python (and C++) was because Python speifically had both a wider range of syntax expressions and more importantly a better optimizability. This allowed for rapid flexible development and deployment, in addition to scalability and performance at an improved maintainability over PHP.

Similar stories exist for so many other platforms, as languages evolve and the second system effect comes into play we will naturally see software evolve to adapt to new requirements and technologies as developers realize their mistakes of the past. Though the rewriting process does not come without its own set of challenges. It could be argued that a contributing factor to the downfall of the Netscape browser was the decision to do a ground up rewrite of the browser, a reason for this being that the original code base was so messy developers did not want to contribute Pogue [1]. The choice was then made to start from new, but by the time the browser released it still lacked a lot of basic features Zawinski [2].

More broadly speaking the manner in which we choose to rewrite software in a different language should preferably align with the core motivation as best as we can, while mitigating as many of the downsides as possible. This is where the concept of language interoperability could come into play. Language interoperability can potentially offer some very interesting benefits, a key one being the ability to effectively implement a gradual/iterative rewrite while still allowing for the original code to be used. The benefit here being that we can leverage the strengths of the new language, still use existing familiar infrastructure, all while allowing the system to be iteratively improved without there being any major downtime untill the rewrite is complete.

2 BACKGROUND

Language interoperability; or interop for short; is broadly seen as the ability of two or more different programming languages to interact with each other and function as a part of a single system. A primary motivation behind the implementation of interop mechanics is the idea that while programming languages; especially modern ones; are often designed to be somewhat general-purpose, they are not necessarily the best tool for every job. The most established form of interop is the Foreign Function Interface (FFI), which often takes the form of a library written in the host language that allows it to call a function via a shared library written in the target language, commonly this being C or C++.

FFIs in cases where there is little mismatch in the runtime and memory models of the host and guest language provides a generally effective means of interop. Though as the languages become more and more differen't we start to see an increase in the overhead with the main contributors to this generally being data marshalling and mismatched execution paradigms. Additionally a key drawback of FFIs are that they generally operate specifically between two languages, while in many cases this is sufficient or even intentional it lacks the flexibility of projects which wish to incorporate more languages. Recent developments in virtual machines especially ones which target ASTs have offered some interesting approaches to solving alot of the existing drawbacks of FFIs.

Meta-JIT Compilers offer some very interesting insights into the future of language interop. In the last 20 or so years we have seen substantial developments in the creation of Meta-JIT compilers, which broadly speaking allow programmers to implement language interpreters, these interpreters are then combined with JIT compilers which either apply tracing; as in the case of RPython; or partial evaluation; as in the case of GraalVM. We can further

divide these Meta-JIT compilers into ones which operate bytecode and ones that operate on ASTs. Whereas RPython can only operate on bytecode GraalVM can both operate on ASTs through its truffle language implementation framework and on bytecode through specifically its llvm language implementation, though at the cost of some optimization ability and interoperability.

A key thing which is so revolutionary with Meta-JIT Compilers in the case of interop is that when combined with some language-agnostic communication interface like the Truffle OSM not only do you have a simple way of implementing a language and having it run on a highly optimized JIT compiler, but because we have everything essentially operating on the same runtime it massively simplifies alot of things that FFI has troubles with. Namely allows for effective language interop in a high performance context with close to no-boilerplate.

Though despite impressive implications the GraalVM has for language interoperability it still has some hurdles that hinder it from a larger scale adoption. One of the biggest ones being that while it does simplify a language implementation, there are substantial compatability limitations with existing infrastructure that will naturally occur. While it currently does support a relatively wide range of language implementations, which have even demonstrated improved performance over their traditional counterparts, it still lacks in compatability, clear developer documentation, and overall developer experience if a language implementation does not already exist.

3 RESEARCH METHODS

To examine the viability of FFI's in the context of testing the performance of a ray-tracing component in different languages a basic ray-tracing system was implemented first entirely in Julia. The most performance critical component was then isolated through profiling and implement in C++ and Python Jluna and Juliacall as the respective interop APIs. With this case study I aim to assess 3 key questions.

- How can we effectively test the performance of a specific component of a system rewritten in another programming language?
- How easy are language APIs to use when rewriting a larger component?
- What are the performance characteristics of interop APIs with different types of guest languages?

3.1 Core Ray Tracer

The core system studied here was a ray tracer implementation in the Julia programming language based in part of RTI1W and PBRT. A ray tracer implementation offers some interesting challenges in the context of language interop. Firstly despite this implementation still being towards the minimal side of things it is complex enough to demonstrate interop in a uniquely applied setting. Additionally a nice benefit in this instance is that we have relatively distinct components, while this does not necessarily transfer to all types of systems it does in some ways mirror the Microservices architecture which is realtively common . Finally ray-tracers especially in the context of real time ray-tracing generally have the requirement of being highly performant, I believe this requirement will give some nice insights into how FFI act under this type of environment.

3.2 Isolation and Rewriting

So asses the above stated research questions I reasoned that in this instance if a certain language did not demonstrate good performance characteristics for a critical component of the system then rewriting the entire system in this language would likely not be a very effective approach. To isolate the critical component I looked at the callgraph from the `pprof` file and then allocations using the `--track-allocations=user` flag when executing the basic rendering for a scene. The choice of C++ and Python would allow for comparing interop between 2 very different languages while being able to use the original julia implementation as the baseline.

3.3 Ease of use and Performance

4 IMPLEMENTATION

4.1 Python Component

4.2 C++ Component

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

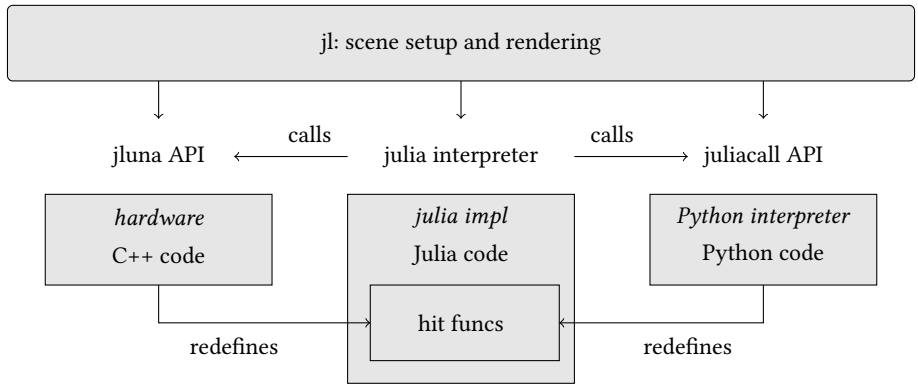

Figure 1: ***System architecture of multi-language PBR renderer.***

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

5 DISCUSSION

Here you put your results in context (possibly grouped by research question). Usually, this section focuses on analyzing the implications of the proposed work for current and future research and for practitioners.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu

wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

6 RELATED WORK

Describe here scientific papers similar to your thesis work, both in terms of goal and methodology. One paragraph for each paper (we expect about 5-8 papers to be discussed). Each paragraph contains: (i) a brief description of the related paper and (ii) a black-on-white description about how your work differs from, or overlaps with, the related paper, hence emphasizing the novelty contributed by this thesis. You may place this section immediately after the Background section, if necessary.

Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero.

7 CONCLUSION

Briefly summarize your contributions, and share a glimpse of the implications of this work for future research.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

REFERENCES

[1] D. Pogue, “State of the art; netscape 6 browser: Mixed bag”, *The New York Times*, Nov. 2000, See the article in its original context from November 30, 2000, Section G, Page 1. [Online]. Available: https://www.nytimes.com/2000/11/30/technology/state-of-the-art-netscape-6-browser-mixed-bag.html.

[2] J. Zawinski, *Resignation and postmortem*, © 1999 Jamie Zawinski, Apr. 1999. [Online]. Available: https://www.jwz.org/gruntle/nomo.html.