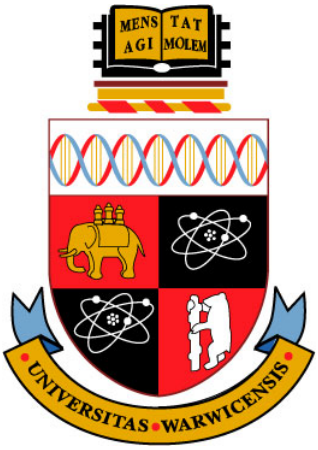

Just In Time Compilation for a High-Level DSL

Nathan Dunne

1604486

3rd Year Dissertation

Supervised by Gihan Mudalige

Department of Computer Science

University of Warwick

2019–20

Abstract

TODO

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.

Key Words

High Performance Computing, Unstructured Mesh,
Just-In-Time Compilation

Contents

List of Figures

1 Introduction

In the field of High Performance Computing (HPC), computers with processing power tens or hundreds of times greater than conventially available machines are used to solve (or appoximate solutions to) problems that would otherwise take an unwarrantable amount of time. Such computers have been required for some time to make use of a large degree of parallelism in order to complete with reasonable runtime: dividing work into independant subsections which can be executed simultaneously.

Many paradigms for executing parallel workloads have emerged over time: including vector instructions (SIMD), many and multi-core CPUs, clusters of interconnected computers, and General Purpose Graphical Processing Units (GPGPUs). Hardware which was originally specilised for graphical shader calculations through it's very high number of processing units, allows carrying out the same operation across a very large amount of data in parallel. This harware has been adapted in GPGPUs to perform non-specific operations that would normally have been done by the CPU.

Furthermore, a large proportion of HPC workloads involve approximating Partial Differential Equations (PDEs) to simulate complex interactions in physics problems, for example the Navier-Stokes equations for computational fluid dynamics, prediciting weather patterns, or computational electro-magnetics. It is necessary to discretise such problems across some form of mesh, either structured (regular) or unstructured. Unstructured meshes will be discussed further in Section 1.1: Background Work.

1.1 Background Work

The OP2 library is an Open Source Domain Specific Langauge (DSL) which provides a high level abstraction for describing physics problems which can be abstracted to an Unstructured Mesh. Unstructed Meshes, such as Figure 2, use connectivity information to specify the mesh topology. The position of elements is highly arbritrary,

unlike structured meshes where elements follow a regular pattern (Figure 1). A particular simulation might, for example, be approximating the velocity of a fluid in each cell based on the cells around it.

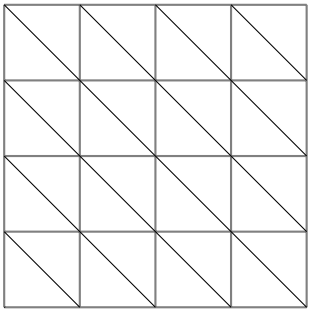

Figure 1: Tri-Structured Mesh

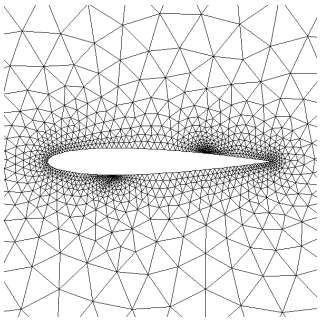

Figure 2: Airfoil Tri-Unstructed Mesh

The abstraction provided by OP2 allows scientists and engineers to focus on the description of the problem, and seperates the consideration for parallelism and efficiency into the back-end library. This is beneficial as it is unlikely that a single developer or team has the necessary expertise in both a niche area of physics with a non-trivial problem to be solved; and sufficient depth of knowledge in computer science to understand and utilise the latest generation of parallel hardware.

A further benefit is that existing solutions which make use of the OP2 library to be ported onto a new generation of hardware, by modifying only the OP2 backend library, and not every application. This portability can save both time and money in development if multiple different hardware platforms are desired to be used.

The OP2 library is already able to generate optimised code for a number of platforms, including the increasingly popular NVidia CUDA for parallel programming on NVidia GPGPUs. However, there is always space for further benefit to be gained, and this report details an investigation into applying a new optimisation to the CUDA code generation library subsection of OP2. The optimisation is named "Just-In-Time Compilation" for its similarities to a comparable process often performed by compilers when runtime efficiency is desired.

1.2 Motivations

The idea for this project was provided by my supervisor, Dr Gihan Mudalige - an Associate Proffesor in the University of Warwick Computer Science Department. I selected it as it aligned with my interest in High Performance Computing, and similar experience with optimising exisiting codes.

Since OP2 is Open Source and freely available, the implementation I produce will become part of the library, allowing future contributors to build on my work. The project also allows me the opportunity to operate on a large codebase, where most university work is done largely within the confines of one's own code.

2 Related Work

3 Specification

4 Implementation

5 Testing

6 Evaluation

7 Future Work

8 Conclusion

Appendices

A example

Acknowledgements

TODO