EXTENDING JUST-IN-TIME COMPILATION FOR OP2

Nathan Dunne - u1604486

October 9, 2019

Problem Statement

OP2 is an Embedded Domain Specific Language for solving unstructured mesh based applications. It exists to provide a level of abstraction to scientists and domain application developers, allowing utilisation of the high performance benefits of advancements in hardware, without requiring them to maintain a high level of knowledge on any new technologies and architectures. As the high-level application written using the OP2 API no longer requires any platform specific optimisations, it can be used on any platform supported by OP2; without the time and financial costs of rewritting the application with different hardware optimisations.

One way to improve the performance of an application such as the unstructured meshes targetted by OP2 is to use Just-In-Time (JIT) compilation, whereby performance sensitive sections of the application are re-compiled at runtime to apply additional optimisations once the parameters of execution are known. The performance gain must be sufficient to offset the extra time taken to recompile in order for this technique to be effective. There is exisiting work implemented already for some JIT optimisations, however there is still space for further performance gain through further parallisation by targetting GPU architecture. This will require generation of cuda intrinsics at runtime. The aim of this project is to implement additional optimisations in this area, such as replacing library functions and other slow operations with constants where possible, hard coding loop bounds, and applying loop fission/fusion at runtime.

There are a number of High Performance Computing applications already written in the OP2 api, from smaller example projects to production applications. Since the API itself will not be altered, it will be easy to apply my additions to the OP2 translator, and determine whether there is performance benefit.

Project Goals

- To gain an understanding of the existing work in the OP2 library, and how it benefits unstructured mesh applications.
- Apply my knowledge of optimisation techniques to implement and test further JIT optimisations.
  - A number of optimisations are possible, so the time to implement each one will determine how many can be included in the project
  - The primary optimisation will be replacing time consuming operations with constants where possible. Ideally the project will also extend to the optimisation of loops.
- To benchmark the performance of existing OP2 applications with and without the additions, and determine whether it provides benefit.

Methods

Research

In order to contribute to the OP2 project, I will first read a number of papers on the op-dsl homepage to gain an understanding of the purpose and implementation of the existing work, and ensure I am able to provide useful addition to the project. Some of the papers suggested are listed in the Further Reading section. Also, In order to increase my familiarity with the exisiting work I will initially be reproducing the exisiting sequential JIT work. This should put me in a comfortable position to begin contributing to the project. Also, as the content of my addition will require good knowledge of the cuda toolkit for GPUs, I will spend some time improving my familiarity with cuda.

Implementation

The implementation of GPU optimisations will require modification so that code is regenerated, probably by OP2's Python generator, then recompiling the newly generated code before it is executed.

The allocated time for this stage will be divided into four rounds of development. Each round will be followed by some detailed testing and analysis of the work completed, which will determine if the next round will be further work on the same optimisation, or beginning to implement a further optimisation. I have selected this process as I am as yet unsure of the complexity of what I intend to implement.

Version Control

The OP2 project is Open Source, and hosted on github[1], so I will work on a branch of the remote repository. Hopefully the completed work will be merged once it is complete.

Benchmarking

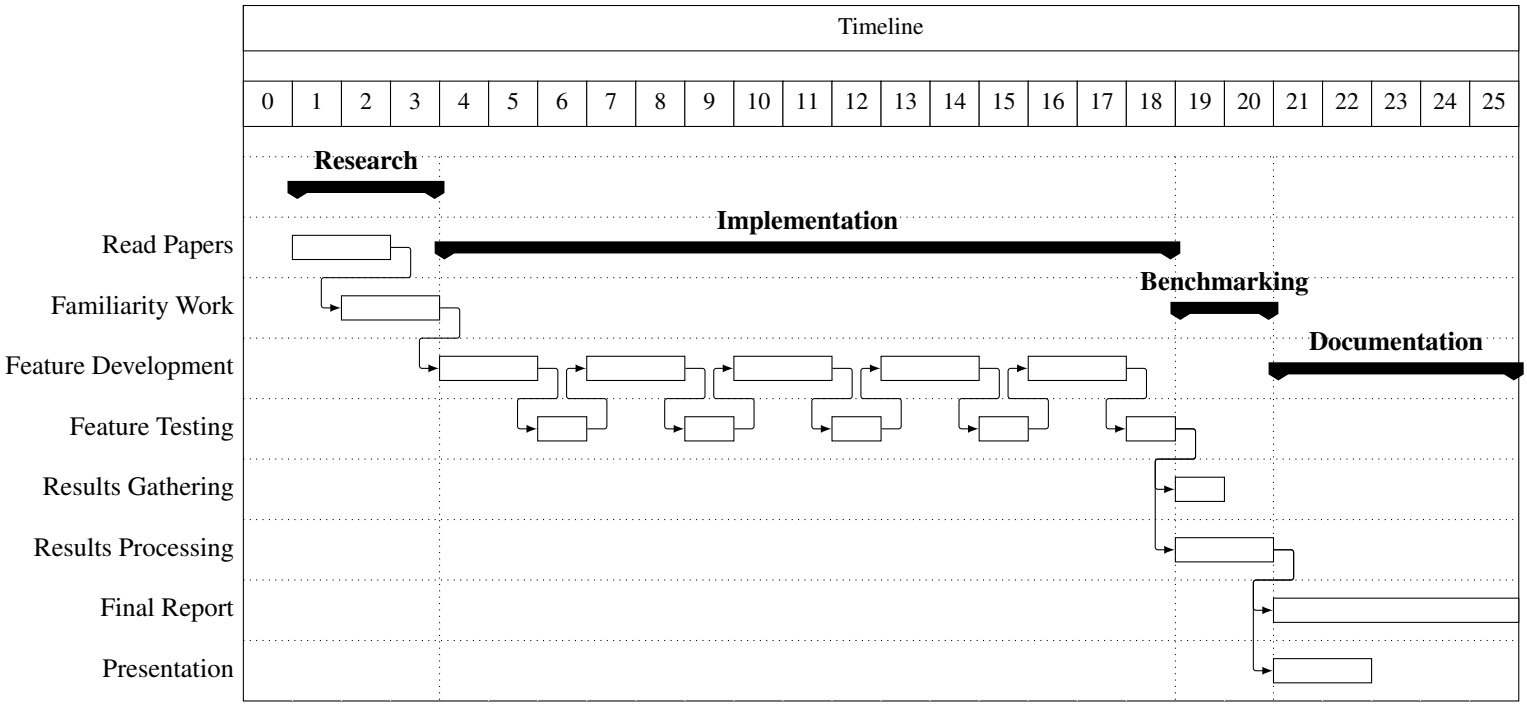
Initally and throughout development I will be using the airfoil[2] example program written using the OP2 api as a testbed for my work. This should be small enough as to be easy to understand, but not so simple that it does not adequately exercise my additions to the OP2 library.
For more rigorous benchmarking, more representative of the expected use case, there are a number of more complex OP2 applications available to assess what performance benefits can be provided.

Documentation

Throughout the project I will keep notes of progress, setbacks and modifcations to the project. Combined with the git history of versions this should mean writing a report on my process will be fairly simple.

Timetable

The timetable for the project is described below. The sections above have been divided into 25 slots of approximately 12 hours of work. This corresponds with there being 28 weeks until the submission of the final report, to allow 12 hours of work a week, and 3 weeks of contingency.



Resources

For the majority of the development work I will be using my personal laptop, however as any changes will be pushed to a remote repository I will be able to work on it from any desirable workstation. This also alleviates concerns of losing work from any failure in my device. In order to complete the cuda sections of the project I will need access to a suitable graphics card. My laptop does have an NVidia graphics card, but should it prove insufficent I will have access to a remote machine to use instead. There is also possibility of extending the JIT capabilities to the ARM aarch64 architecture, in which case this will also require remote hardware.

Potential Issues

I do not see any potential ethical or social issues, as the project will not require the collection or storage of any personal data. There is however a legal consideration of the lisence under which the open source project is held. The 2-Clause BSD license permits redistribution of source and binary, as long as it contains the copyright disclaimer, so I will ensure to uphold this requirement.

References

[1] Op2-common. https://github.com/OP-DSL.

[2] Istvan Reguly Mike Giles, Gihan Mudalige. Op2 airfoil example. 2012. https://op-dsl.github.io/docs/OP2/airfoil-doc.pdf.

Further Reading

[3] I.Z. Reguly G.R. Mudalige and M.B. Giles. Auto-vectorizing a large-scale production unstructured-mesh cfd application. 2016. https://www.oerc.ox.ac.uk/sites/default/files/uploads/profile-pages/Gihan/GRM-WPMVP.pdf.

[4] M.B. Giles G.R. Mudalige, I. Reguly. Op2: An active library framework for solving unstructured mesh-based applications on multi-core and many-core architectures. 2012. https://www.oerc.ox.ac.uk/sites/default/files/uploads/profile-pages/Gihan/InPar2012.pdf.

[5] I.Z. Reguly et al. Acceleration of a full-scale industrial cfd application with op2. 2015. https://www.oerc.ox.ac.uk/sites/default/files/uploads/profile-pages/Gihan/OP2-Hydra.pdf.