Contributing to Futhark for your Bachelors Project

Troels Henriksen (athas@sigkill.dk)

Computer Science
University of Copenhagen

3rd of September 2018

Contribute to the best data-parallel GPU-targeting ML-like functional language developed at DIKU!

Looks a bit like a simplified combination of SML and Haskell.

```
let vector_add [n] (a: [n]i32) (b: [n]b): [n]i32 =
  map2 (+) a b

let product [n] (a: [n]i32): i32 =
  reduce (*) 1 a

let dot_product [n] (a: [n]i32) (b: [n]i32): i32
  product (vector_add a b)
```

It runs *very fast* on parallel hardware. Significantly faster than most hand-written C or similar.

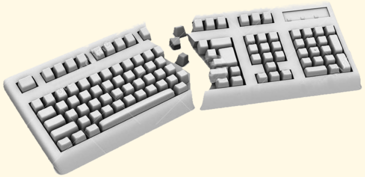
Roughly Three Kinds of Projects

(Light) Compiler Hacking:
The Futhark compiler is written in Haskell. It is well written, but not small (55k SLOC). Only fairly limited work possible for a bachelors project.

Parallel Programming in Futhark:
We port benchmarks, libraries, and example applications to determine areas in which Futhark should be improved.

Work on Infrastructure and Tooling:
Futhark is run as a sound software engineering project, which means tooling to run tests, analyse benchmark results, etc. These are very "open" projects, in that you are free to choose language and methodology yourself.

The Unit of Difficulty



- We rate the difficulty of project proposals from one to five broken keyboards.
- Not simply a measure of workload, but based on whether we believe we understand every issue related to implementation of the project.
- Higher difficulty implies more "unexpected" problems that we cannot immediately give you an answer to.

Project: Java Backend

Idea The host code could be generated in any language with OpenCL bindings. Since most of the runtime is ideally spent in the GPU code, performance of the host language should not matter much. We already have code generators for Python and C, and soon C# - we would like to add Java to this list.

Challenges Mapping Futhark IR constructs and types to Java; generating a nice API. However, you will have to write code in Haskell.

Difficulty

⌨ ⌨

Prior generations of bachelors students have already found the landmines for you - but it's still fun and rewarding work.

Project: Java Backend (details)

In principle, we could have a backend code generator for every language out there. A fully functioning code generator is not particularly large (less than 2000 SLOC for the Python backend). However, they are still a maintenance burden, so we would like to keep the number limited. JVM and .NET are, however, interesting, because by supporting these, Futhark would immediately be accessible to all other languages running on these platforms (Java, C#, Kotlin, F#, Scala, etc).

One thing that eased the development of the Python backend (which was itself a bachelors project in 2016) was the presence of a mature library for calling OpenCL. These also exist for .NET (either NOpenCL or OpenCL.Net), and the JVM (JOCL).

This project will require you to write a nontrivial amount of code in Haskell, although the existing backends provide a pretty good template to follow. For example, see the following files:

- `https://github.com/diku-dk/futhark/blob/master/src/Futhark/CodeGen/Backends/GenericPython/AST.hs`
- `https://://github.com/diku-dk/futhark/blob/master/src/Futhark/CodeGen/Backends/GenericPython.hs`
- `https://github.com/diku-dk/futhark/blob/master/src/Futhark/CodeGen/Backends/PyOpenCL.hs`

The Python backend is pretty full-featured. We don't expect all the bells and whistles from a bachelors project.

Project: Investigate and Improve C# Backend and F# Frontend

Idea	We just finished a succesful master's thesis project that implemented a C# backend for Futhark. This project also includes a compiler from a small subset of F# to Futhark. We think it works well. We might be wrong. We'd like someone to investigate more closely.
Challenges	You'll be working with fairly new technology that is pretty untested, even by the standards of research language.
Difficulty	⌨ It is very easy to scale this project based on your ambitions, skills, and interests.

Project: Investigate and Improve C# Backend and F# Frontend (details)

The C# backend can produces standalone executables, just like the other backends. This has been tested quite well, but is useful only for benchmarking. The ambition of the C# backend is that the generated code can be accessed by otherwise ordinary C# applications, but this has not been evaluated much. This project could investigate how easy it is to take existing C# programs with some computational component, and rewrite that component in Futhark. Alternatively, it could take one of the existing Futhark demos[1][2][3], which are written in Python and Go, and rewrite them in C#.
We also have a compiler, FShark, from a subset of F# to Futhark. The point here is that an F# programmer can develop and debug F# code as they are used to and with conventional tools (subject to some extra rules), and then later compile it to Futhark, and from there to efficient GPU code. We have no idea how practical this workflow is. Maybe you can find out for us?

The overarching purpose of this project is to produce feedback for improving these two bits of technology. If you

are up to it, the project can also involve actually *performing* said improvement, but it is not a requirement.

[1]https://github.com/Athas/diving-beet
[2]https://github.com/Athas/futball
[3]https://github.com/nqpz/futcam

Project: CUDA Backend

Idea Currently, the Futhark compiler generates code using the OpenCL library, but some interesting platforms support only NVIDIAs CUDA. It would be useful to add a CUDA backend to Futhark.

Challenges There are three different CUDA layers that could be targed by the code generator: language, runtime, and driver level. I have no idea which one is best. Will possibly require some minor modification of kernel code generator (the CUDA and OpenCL kernel languages are similar but not identical).

Difficulty

⌨ ⌨ ⌨ ⌨ ⌨

Project: CUDA Backend (details)

This project involves a significant amount of independent work on studying the CUDA documentation and determining how to use its low-level facilities correctly.

In order to ease maintenance, the new CUDA backend should re-use most of the concepts used by the existing OpenCL backend. Hence, you will probably have to write some glue code to make CUDA look a bit more like OpenCL. I do not expect this to be conceptually difficult.

You will need to write a fairly nontrivial amount of Haskell for this one. However, you will be working at the very tail end of the compiler, so you will not have to spend a lot of time studying an existing large code base. In a perfect world, you might be able to get away with simply writing some runtime system code (in C/C++) and a CUDA-fied version of this module—

`https://github.com/diku-dk/futhark/blob/master/src/Futhark/CodeGen/Backends/COpenCL.hs`

—but we do not live in a perfect world.

One very interesting question is whether code generated by the CUDA backend will actually run faster than the equivalent code generated by the OpenCL backend, and if so, why.

Project: Implement Language Server Protocol (LSP) for Futhark

Idea *"The Language Server protocol is used between a tool (the client) and a language smartness provider (the server) to integrate features like auto complete, go to definition, find all references and alike into the tool"*. Implement an LSP server for Futhark to get these features in various code editors that support LSP.

Challenges The LSP is a real industrial spec, and probably too large to implement completely in a single bachelor's project. You will have to identify a realistic yet practically useful subset.

Difficulty

⌨ ⌨ ⌨

Project: Implement Language Server Protocol (LSP) for Futhark (details)

Futhark already has an Emacs major mode that supports syntax highlighting, (some) automatic indentation, and such. We could work on adding support for autocompletion and *go to definition* functionality in an ad-hoc way, but not only is it a lot of work, it would also not be accessible to those unfortunate people who do not use Emacs. By implementing an LSP server instead, this functionality would also be available to people using such editors as VSCode, Sublime Text, Vim, etc.

The Futhark compiler frontend can be used to analyse a Futhark program to obtain identifiers and the locations of various definitions. Realistically, in order for this information to be accessible, the LSP server will have to be written in Haskell. Alternatively, I could extend the Futhark compiler to dump all this information in some standard format (JSON) which you could read in an LSP server implemented in another programming language.

While I have not ever implemented an LSP server myself, it is my impression that it is not overly complicated. However, this is definitely a coding-heavy project.

The LSP spec	`https://microsoft.github.io/language-server-protocol/specification`

Other Projects

If you have an idea of your own, come by and we can have a chat about it. Inspiration:

- A "Try Futhark" web application like `https://tryhaskell.org/`.
- Port a benchmark from a suite like Rodinia or Parboil to Futhark.
- Writing a Futhark package or application that does something cool: classification, machine learning, image analysis, etc.
- An automatic code formatter like `gofmt`.
- Fixing/improving/rewriting the automatic indentation of the `futhark-mode` for Emacs.

If you have an ≤ 15 ECTS idea for something that would contribute to the project, we can probably supervise it.

Compiler hacking is hard work, but rewarding

“Futhark: the harder the battle the sweeter the victory!”
-Rasmus Wriedt Larsen (bachelor and masters thesis)

“Hvis du laver et Futhark-projekt, kommer du aldrig til at sove.”
-Niels G. W. Serup (bachelor and masters thesis)

“Et ambitiøst projekt med en hårdt arbejdende kerne af folk! Hvad mere kan man ønske sig?” -Hjalte Abelskov (bachelor thesis)

“Efter at havde arbejdet med Futhark føles alt andet i mit liv meningsløst.” -Daniel Gavin (bachelor thesis)

“Futhark: Så lille et sprog, der kan så meget for så mange!”
-Kasper Abildtrup Hansen (bachelor thesis)

Contact

If you think any of this is interesting, come talk to Troels...

- ...in the office shared with Cosmin: 01-0-017 at HCØ.
- ...via email: `athas@sigkill.dk`
- ...or on IRC: #diku on Freenode (`http://ucph.dk/#tabs1-chat`)

Or talk to Martin Elsman or Cosmin Oancea.

Also check out the website at `https://futhark-lang.org`