CSE211: Compiler Design

Homework 4: Domain Specific Languages
Assigned: Nov. 29, 2023
Due: Dec. 15, 2023

Preliminaries

This assignment is more free-form than the previous assignments.

1. Please read the instructions carefully. Be creative and have fun!
2. This homework is designed for pairs. Use pair collaboration techniques: both partners should be involved in all parts of the work. If you're working remotely, share your screen at all times. Use this opportunity to help each other understand the material. Discuss and fully understand the concepts before moving on.
3. If you face any issues with your partner, inform us as soon as possible.

1 Exploring a DSL

In this assignment, you will pick one (out of three) DSLs to explore. Your exploration will consist of the following:

- Reading the original DSL academic paper
- Obtaining the code for the DSL
- Running experiments with the code
- Recording your experiences and results in a report

My understanding is that all of these are well-supported and maintained, at least for Linux. I am not sure about other systems. Because of this, you may want to develop on the docker image.

1.1 Pair Programming

Even though the deliverable for this assignment is a report, it is still a pair programming assignment. Please do the programming component together. Once that is completed, please share the responsibility of writing equally (although you do not have to be together when the report is written).

2 The DSL options

2.1 Halide

Halide is one of the most influential modern DSLs. Originally written for image processing, it's ideas on abstracting computation from optimization have been used widely.

- The paper for Halide is here: `https://people.csail.mit.edu/jrk/halide-pldi13.pdf`
- The code for Halide is here: `https://github.com/halide/Halide`
- A tutorial for Halide is here:
  `https://halide-lang.org/tutorials/tutorial_introduction.html`

If you choose Halide, please work your way at least through tutorial number 8.

2.2 GraphIt

GraphIt is a more recent DSL for optimizing graph computations, similar to the flow analysis that we studied in module 2. It allows the user to specify a vertex-centric program and the backend implements optimizations such as load balancing and graph traversal direction.

- The paper for GraphIt is here: `https://dl.acm.org/doi/pdf/10.1145/3276491`
- The code for GraphIt is here: `https://github.com/GraphIt-DSL/graphit`
- A tutorial for GraphIt is here: `https://graphit-lang.org/getting-started`

The entire tutorial should be possible for GraphIt

2.3 TVM

TVM is a machine-learning DSL. It operates at a lower-level than the usual suspects in this domain (TensorFlow and PyTorch), however, its backend is able to fuse DNN operators and specialize to different input shapes more generally than the TensorFlow and Pytorch tools.

- The paper for TVM is here: `https://arxiv.org/pdf/1802.04799.pdf`
- The code for TVM is here: `https://github.com/apache/tvm`
- A tutorial for TVM is here:
  `https://tvm.apache.org/docs/tutorial/autotvm_matmul_x86.html`
  and
  `https://tvm.apache.org/docs/tutorial/auto_scheduler_matmul_x86.html`

If you choose TVM, you should work your way through both tutorials.

2.4 Other DSLs?

If you would like to try out a DSL that isn't on this list, please propose the DSL to me no later than one week from the assigned date. In order to be approved, the DSL should have an associated academic paper, well-maintained open-source availability, and an accessible tutorial. There is no guarantee that I will approve a DSL, even if it has all of the above criteria, so please plan accordingly.

3 Report

Your report will be 6 pages double spaced in the default MS word or Google Doc format (size 11 font). If your margins, spacing, or font is selected in such a way that your word count is significantly lower than what it would be in the default configuration then you will be liable to lose points. This is a grad class and I expect a corresponding level of quality in your work. You are free to go over the page limit if you wish.

The two page should be a summary of the DSL, including it's domain, shortcomings of existing languages in the domain, and its key ideas for optimizations and expressiveness.

The next two pages (roughly) should be an experience report (along with your opinion) of the DSL in terms of its qualitative features. For example: do you think the language is given at the right level of abstraction? What are some of the trade-offs that the language made in terms of specialization vs. general computation? Was it easy to install and run the tutorial programs? What does it do well? How do you think it could improve? Are programs easier to reason about? Are common bugs avoided by the language constraints?

The final two pages (roughly) should be your experiences running some programs on the DSL. Try to run enough programs so that you can showcase as many of the optimizations as possible. Then report on your programs and the performance impacts of the optimizations. You can (and should) include other experiments here. For example: Try running the autotuner for the optimizations (if available) and comment on the parameters it finds. Try running the programs with different input types and shapes. For example, if the DSL takes a 2D data structure, try it with tall/skinny data, and with short/wide data and comment on if different parameters work better or worse for different input types. For GraphIt, if it is available you should try on different graphs, e.g. road networks and social networks.

While it isn't required, you could compare the performance of some of the optimized DSL codes to other implementations. For example, TVM and Halide programs can be written in Numpy (python). Are TVM and Halide programs faster or slower than similar Numpy programs? This will be harder for GraphIt, but it might be possible to compare against Professor Beamer's GAPS graph benchmarks: `https://github.com/sbeamer/gapbs`.

It may be interesting for those of you with different systems (e.g. M1 and x86) compared results to see if different optimizations were more or less effective across the different systems.

If the DSL has a GPU backend, and if it can target your GPU, you may want to try that as well.

Be creative and have fun!

3.1 Data and images

For your report: please try to use all original images and data. If you use data or images from anywhere else, please cite it.

Please make your images only as large as they need to be. Only 1.5 pages of images should be counted towards your 6 page limit. Feel free to go over the limit if you'd like.

4 Considerations

A few final considerations:

- Use best practices when obtaining experimental results. For example, use an input such that you can obtain meaningful (i.e. long enough) running times. Run several times and take an average. If there is high variance between runs, please report it.
- Feel free to post on Piazza if you need help with getting the tools running. It is fine to work together on infrastructure tasks. It is also fine to ask questions and discuss the papers.