Scalable Precision Tuning of Numerical Software

(the slides are available under “Presentation Materials” in the above URL)
Date: October 14, 2020
Presented by: Cindy Rubio González (UC Davis)

Q. Is there a possibility of using these tools with TensorFlow or Pytorch?

A. The techniques should be applicable to any language, but the implementations discussed here do not work on Python code.

Q. How expensive is it to run the tool?

A. The runtime of the tool depends on (1) how large the search space is, and (2) how long it takes to test each program configuration explored by the algorithm.

Q. Will the Fortran support you talked about rely on Flang?

A. No. Unlike the work described today, the Fortran support works at the source level and does not rely on Flang.

Q. Would the tool support templated types (in C++) ?

A. The current implementation has mainly been tested on C code. We would need to inspect the code transformations that involve templates to add them to the set of transformation rules.

Q. What are the advantage/challenges of working on the source code compared to the LLVM IR (apart from removing the dependency on a specific toolchain ?)?

A. Working on the LLVM IR facilitates a wider variety of program transformations that would be difficult to apply at the source level. However, in terms of the transformations described today, I believe there is not a large advantage of working on LLVM IR over source code.

Q. If we have a C++ or C code to run on some exotic hardware (ex. Cerebras CS1) could we intercept the IR and apply these tools? How do we give the optimizer the numeric output and timings?

A. We need a way to create an executable for the desired hardware. The current implementations use LLVM, thus we would need to be able to compile the programs with the clang compiler. Both Precimonious and HiFPTuner require as input the error threshold to use during tuning. Furthermore, the program needs to be annotated to include some calls to utility functions that will log the results each time a transformed program is run, check whether the result is within the given error threshold with respect to the original result, and measure runtime.

Q. Do these tools support precision tuning on GPUs or mixed CPU/GPU systems?

A. No, but recent work has applied similar approaches to perform precision tuning of GPU applications, e.g., GPUMixer.

Q.Is there a way to only apply these technologies to part of the code? For example, for a pre-conditioner I would be willing to give up precision, but I don't want to reduce accuracy for the non-linear solve? I might have missed that in the talk. Or would I just create a code that just calls the pre-conditioner by itself?

A. Yes, Precimonious and HiFPtuner allow you to provide specific functions/variables to consider during precision tuning, or functions/variables to exclude from the search space. There is no need to create code that would just call the parts you want to tune, though if that reduces the running time considerably, then it would be a good option to speed up the search.